Reasoning with Sorted-Pareto Dominance and other Qualitative and Partially Ordered Preferences in Soft Constraints

Conor O'Mahony
BSC MSC
96598964

NATIONAL UNIVERSITY OF IRELAND, CORK

FACULTY OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CORK CONSTRAINT COMPUTATION CENTER (4C)

Thesis submitted for the degree of Doctor of Philosophy

4th October 2013

Head of Department: Prof Barry O'Sullivan

Supervisors: Dr Nic Wilson
Prof Barry O'Sullivan

Research supported by Science Foundation Ireland

Contents

I, Conor O'Mahony, certify that this thesis is my own work and I have not obtained a degree in this university or elsewhere on the basis of the work submitted in this thesis.

Conor O'Mahony

To Mam and Dad - thanks for everything.

Acknowledgements

I would like to acknowledge the help and support of the following people.

Firstly, I would like to thank my primary supervisor Nic Wilson for all the guidance, support and patience, from the very beginning of this journey right to the very end – I appreciate all and everything that you have done for me during the last few years. I would also like to acknowledge and thank my secondary supervisor Barry O'Sullivan, and also Science Foundation Ireland for the financial support.

I would like to thank all the reviewers of my papers submitted to different workshops and conferences, for the constructive criticism and pointers that helped me improve and write better papers. I would like to thank my internal examiner Ken Brown and my external examiner Gérard Verfaillie for their comments and feedback and for the enjoyable discussion during my viva voce.

I would like to thank the 4C lab admin office, Eleanor, Caitríona and Linda for all their help handling all of the administration of the thesis, and also the 4C IT department Joe and Peter for handling all my IT needs.

Finally, I would like to thank my partner Adele, my parents Eileen and Diarmuid, my siblings Sandra Eithne and Liam, and my extended family and friends for all their support.

Many thanks to all of you,

Conor.

Abstract

In decision making problems where we need to choose a particular decision or alternative from a set of possible choices, we often have some *preferences* which determine if we prefer one decision over another. When these preferences give us an ordering on the decisions that is complete, then it is easy to choose the best or one of the best decisions. However it often occurs that the preferences relation is partially ordered, and we have no *best* decision. In this thesis, we look at what happens when we have such a partial order over a set of decisions, in particular when we have multiple orderings on a set of decisions, and we present a framework for qualitative decision making. We look at the different natural notions of *optimal* decision that occur in this framework, which gives us different optimality classes, and we examine the relationships between these classes. We then look in particular at a qualitative preference relation called Sorted-Pareto Dominance, which is an extension of Pareto Dominance, and we give a semantics for this relation as one that is compatible with any order-preserving mapping of an ordinal preference scale to a numerical one. We apply Sorted-Pareto dominance to a Soft Constraints setting, where we solve problems in which the soft constraints associate qualitative preferences to decisions in a decision problem. We also examine the Sorted-Pareto dominance relation in the context of our qualitative decision making framework, looking at the relevant optimality classes for the Sorted-Pareto case, which gives us classes of decisions that are necessarily optimal, and optimal for some choice of mapping of an ordinal scale to a quantitative one. We provide some empirical analysis of Sorted-Pareto constraints problems and examine the optimality classes that result.

List of Figures

Chapter 2

Chapter 3

Chapter 4

List of Tables

Chapter 4

Chapter 5

Chapter 6

List of Theorems

List of Definitions

Chapter 2

List of Examples

Chapter 1

Introduction

1.1 Introduction

In any given decision making problem, we have a set of *decisions* or *alternatives* or *choices*, and the task may be to choose a decision from this set which is in some way the most preferred one. Considering that we have some preferential information available for each decision, then we can form a *preference relation* on the set of decisions, where for a pair of decisions we can say that we *prefer* one over the other. If we have enough information available then we might be able to compare every pair of decisions, and therefore the resulting relation is *total*. In this case, the decision to choose then is simply the best one, or if we have more than one best decision, i.e., we have a set of equivalent decisions that are better than all the others, then we choose one from this set. However, it can also occur that there is not enough information to do this, and we end up not being able to compare every pair of decisions in the set. This gives us the notion of *incomparability* between decisions, and as a result we have a *partial* ordering on the set, and as such we may no longer have a best decision or set of equivalent best decisions from which to choose.

Partial orders in decision making can occur in many natural situations, for example, in multi-criteria decision making where we are comparing decisions on more than one criteria, or in decision making under uncertainty, where there is more than one possible state of the world to consider. They can also occur when the preferential information available is of a *qualitative* nature, and we cannot numerically aggregate preference values associated to a decision. For example, in the quantitative case, if we were evaluating two decisions based on some cost or utility, then we could add the costs or utilities associated to each decision and compare the sum of these costs, which would give us a total order on the set of decisions. However if we have only qualitative preferential information associated with each decision then we cannot do this.

In partially ordered decision making situations, if we cannot choose a single "best" decision, then is there another way to choose a decision that is in some way better than the other decisions? In some cases, we may not want to choose a single decision, but instead present a set of decisions that are the better ones to another decision maker so that they can choose. Can we classify decisions in such a way that we can see which decisions are better in some way than the other decisions? Might a decision be the best one given some criterion if we are looking at multiple criteria, or given some possible scenario if we are dealing with uncertainty? What other notions of optimality are there? The focus of this thesis is to look at some decision making

situations where we have partially ordered or qualitative preference information, and investigate how we can solve these problems and classify decisions in such a way so that a decision maker can make their choice.

1.2 Overview and Contributions

In this section, we give a brief outline of the main chapters in the thesis, and we also highlight the main contributions.

Chapter 2: Background I - Preferences

In this chapter, we look at some background material on preference relations. We describe different types of preorder relations and look briefly at the relationships between them, in terms of the resulting orderings over a given set of elements, and the elements that are the "“best” elements and the “optimal” elements of the set. We also introduce the Sorted-Pareto dominance relation, which is a preference relation we explore later in Chapters 5 and 6 and we look at some specific relations in different decision making situations that are connected to the Sorted-Pareto dominance relation. We provide some new results and new formulations of existing results which describe these connections between Sorted-Pareto dominance and related preference relations.

Chapter 3: Background II - Soft Constraints

In this second introductory chapter, we give some background material on constraint networks. First we look at hard constraints problems and the basic terminology associated with hard constraints networks. Then we look at soft constraints, which are a generalisation of hard constraints, and we define some of the constraints formalisms that are related to works in the thesis, for example, weighted constraints, which specify weight values to associate with variable assignments in a given weighted constraint problem. We look at a general constraints problem which incorporates both hard and soft constraints, and we look at some methods on how to solve these problems, which include depth first and branch and bound searches.

Chapter 4: Qualitative Notions of Optimality

In this chapter, we look at the first main contribution of the thesis. We consider a qualitative framework for decision making, where for a set of decisions we have multiple ordering or rankings, and we look at the relations for comparing decisions and the resulting notions of optimal decisions that occur, which gives us a classification of notions of optimality. We precisely describe the relationships between the optimality classes and also investigate how these classes can simplify under extra conditions.

Chapter 5: Sorted-Pareto Dominance and Soft Constraints

In this chapter, we look at the second main contribution of the thesis. We consider a qualitative preference relation called Sorted-Pareto dominance, which is an extension of the Pareto Dominance relation. We give a semantics for Sorted-Pareto dominance, as a relation that is compatible with any order-preserving mapping that maps a qualitative scale to a quantitative scale. We look at Sorted-Pareto dominance in a Soft-Constraints setting, where we describe our implementation of a Soft Constraints Solver, and detail three different search algorithms (and variations thereof) for solving Sorted-Pareto problems. We give some experimental results for solving different problem instances, comparing Sorted-Pareto dominance to other preference relations and examining the sizes of the resulting sets of optimal solutions. We also look at an extension to the Sorted-Pareto dominance relation, for minimising the maximum costs of the Sorted-Pareto non-dominated solutions.

Chapter 6: Sorted-Pareto Dominance and Qualitative Notions Of Optimality

In this chapter, we look at the third main contribution of the thesis. The analysis in Chapter 4 suggests different natural notions of optimality which are applicable to the connection between Sorted-Pareto dominance and the Min-Sum of weights relation given in Section 5.3. Using this framework from Chapter 4, we capture this relationship between the Sorted-Pareto dominance relation for qualitative preferences and the sum of weights relation for quantitative preferences. We see how the relationship between the optimality classes from the qualitative framework simplify in this instance, and we look at some experimental results where solutions are classified according to their optimality classes.

Situations that are partially ordered or use qualitative information are important as they occur naturally in decision making, so frameworks and preference relations that deal with these situations are motivation for this thesis. Therefore in this thesis we develop a unifying framework for qualitative decision making which can be used by a decision maker, for example, in the context of decision support or recommender systems, to help narrow down the set of decisions or alternatives from which the decision maker can choose. In this framework, the optimality classes presented in Chapter 4 show how decisions can be categorised according to their optimality class, which can be useful in order to aid a decision maker in their choice. The Sorted-Pareto dominance relation presented in Chapter 5 is a qualitative preference relation for which we show that the non-dominated decisions have semantic significance and are also experimentally useful in the context of selecting decisions to present to a decision maker. Further optimality classes in relation to Sorted-Pareto dominance are developed in Chapter 6 which further add to this overall unifying framework for qualitative decision making.

1.3 Publications

Parts of the work of the thesis have appeared in the following published proceedings of various conferences and workshops, which have been subject to peer review.

Conference Papers

[OW13] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance and qualitative notions of optimality.

In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2013,* 2013.

[OW12] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance: an extension to Pareto Dominance and its application in Soft Constraints.

In *Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2012,* 2012.

[WO11] Nic Wilson and Conor O'Mahony. The relationships between qualitative notions of optimality for decision making under logical uncertainty.

In *Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science, AICS 2011,* 2011.

Workshop Paper

[OW11] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance: an extension to Pareto Dominance and its application in Soft Constraints.

In *Proceedings of CP 2011 Workshop: Preferences and Soft Constraints (SofT'11), CP 2011,* 2011.

Chapter 2

Background I - Preferences

2.1 Introduction

In a decision making problem, we have a set of *decisions* (or *options*, *alternatives* or *choices*), and the task is to choose a single decision or subset of decisions that are in some way *preferred* to the other decisions. Aside from the information relating directly to the actual decision itself, for example, the actual domain values assigned to the decision variables in a combinatorial problem, we can have additional *preference* information associated with a decision. This preference information can take various different forms, for example, it could form a collection of evaluations by one or more agents [vN28, vNM47] or with respect to one or more criteria [KR76, Ehr05] where the evaluations represent the cost incurred or satisfaction gained should a decision be realised. Using this preference information, the set of decisions can be ordered or ranked in relation to each other, and this ordering on the set is called a preference relation.

The outline of the Chapter is as follows. In Section 2.2 we describe some different types of preference relations and give definitions of *best* and *optimal* elements, which are the elements of a given set that are *preferred* in some way according to a given preference relation. Section 2.3 looks at preference relations in a general decision making context and looks at what sort of preference information might be available. In Section 2.4, we introduce the Pareto dominance relation, which is a preference relation that compares decisions over multiple preference values, and in Section 2.5 we define Sorted-Pareto dominance, which extends Pareto Dominance, and which is the main focus of Chapter 5. In Section 2.6, we look at some other preference relations and works that are related to the Sorted-Pareto dominance. We consider preference relations that work with qualitative and quantitative preference scales, and we also consider different preference relations that maximise positive preference or minimise negative preferences.

Unless otherwise stated, the basic results in this chapter are well known but to our knowledge do not appear anywhere else in their current formulation.

2.2 Preference Relations

Firstly, for some set $\mathcal{A}$, a *binary relation* R on $\mathcal{A}$ is a set of ordered pairs of the elements of $\mathcal{A}$, i.e., R is a subset of the Cartesian product $\mathcal{A}^2 = \mathcal{A} \times \mathcal{A} = \{(a, b) : a \in \mathcal{A}, b \in \mathcal{A}\}$.

We now define a *preference relation* which is a type of binary relation.

Definition 2.1 » Preference relation
A *preference relation* R on a set $\mathcal{A}$ is a binary relation such that for any $\alpha, \beta \in \mathcal{A}$, if $(\alpha, \beta) \in \mathrm{R}$, then we say that α is *preferred* to β. «

It is also called an *order relation*, as it gives an ordering over the set of elements. Given any $\alpha, \beta \in \mathcal{A}$, if α is preferred to β according to relation R, i.e., if $(\alpha, \beta) \in \mathrm{R}$, we can also write this as $\alpha \,\mathrm{R}\, \beta$. If we have that α is not preferred to β according to relation R, i.e., we have that $(\alpha, \beta) \notin \mathrm{R}$, then we can write this as $\alpha \not\mathrm{R} \beta$. Next, we give some definitions of basic properties of order relations in the following remark.

Remark 2.1 » Basic properties of a relation
A relation R on a set $\mathcal{A}$ is:

(i) *reflexive* if, for all $\alpha \in \mathcal{A}$, $\alpha \,\mathrm{R}\, \alpha$;

(ii) *transitive* if, for all $\alpha, \beta, \gamma \in \mathcal{A}$, if $\alpha \,\mathrm{R}\, \beta$ and $\beta \,\mathrm{R}\, \gamma$, then $\alpha \,\mathrm{R}\, \gamma$;

(iii) *complete* if, for all $\alpha, \beta \in \mathcal{A}$, either $\alpha \,\mathrm{R}\, \beta$ or $\beta \,\mathrm{R}\, \alpha$;

(iv) *antisymmetric* if, for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \,\mathrm{R}\, \beta$ and $\beta \,\mathrm{R}\, \alpha$, then $\alpha = \beta$;

(v) *symmetric* if, for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \,\mathrm{R}\, \beta$ then $\beta \,\mathrm{R}\, \alpha$;

(vi) *asymmetric* if, for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \,\mathrm{R}\, \beta$ then $\beta \not\mathrm{R} \alpha$.

We now look at some specific types of order relations. We consider in particular the four types of preorder relation defined as follows.

Definition 2.2 » Preorder relations
A relation R on a set $\mathcal{A}$ is a:

(i) *preorder* on $\mathcal{A}$, if it is reflexive and transitive;

(ii) *partial order* on $\mathcal{A}$, if it is a preorder (i.e., reflexive and transitive) and antisymmetric;

(iii) *total preorder* on $\mathcal{A}$, if it is complete and transitive;

(iv) *total order* on $\mathcal{A}$, if it is a total preorder (i.e, complete and transitive) and antisymmetric. «

2.2.1 Associated Relations to a Preorder

For a preorder R on a set $\mathcal{A}$, we look at some relations that are associated with R, and which can also be defined in terms of R. First we look at the *strict* or *asymmetric* part of R, which is defined as follows.

Definition 2.3 » Strict relation
For a preorder R on a set $\mathcal{A}$, the associated *strict preorder* relation $\mathrm{S_R}$ is given by

— for all $\alpha, \beta \in \mathcal{A}$, $\alpha \,\mathrm{S_R}\, \beta$, if and only if, $\alpha \,\mathrm{R}\, \beta$ and $\beta \not\mathrm{R}\, \alpha$. «

This relation $\mathrm{S_R}$, is irreflexive and transitive, and represents the notion of strict preference, so for some $\alpha, \beta \in \mathcal{A}$, if $\alpha \,\mathrm{S_R}\, \beta$, then α is *strictly preferred* to β, with respect to relation R. We now look at the *symmetric* part of R, which is defined as follows.

Definition 2.4 » Equivalence relation
For a preorder R on a set $\mathcal{A}$, the associated *equivalence* relation $\mathrm{E_R}$ is given by

— for all $\alpha, \beta \in \mathcal{A}$, $\alpha \,\mathrm{E_R}\, \beta$, if and only if, $\alpha \,\mathrm{R}\, \beta$ and $\beta \,\mathrm{R}\, \alpha$. «

This relation $\mathrm{E_R}$ is reflexive, symmetric, and transitive; it represents the notion of equally preferred, if $\alpha \,\mathrm{E_R}\, \beta$, then α is equivalent (or equally preferred) to β, with respect to relation R. This also gives us the notion of an equivalence class of a particular element $\alpha \in \mathcal{A}$, which is the set consisting of all the elements of $\mathcal{A}$ that are equivalent to α. We define this as follows.

Definition 2.5 » Equivalence class
For a preorder R on a set $\mathcal{A}$, the *equivalence class* of an element $\alpha \in \mathcal{A}$, denoted by $[\alpha]_{\mathrm{R}}$, is defined as

— $[\alpha]_{\mathrm{R}} = \{\beta \,:\, \beta \in \mathcal{A}, \beta \,\mathrm{E_R}\, \alpha\}$ «

When elements cannot be compared with respect to R, we have the following relation.

Definition 2.6 » Incomparability relation
For a preorder R on a set $\mathcal{A}$, the *incomparability* relation $\mathrm{I_R}$, is given by

— for all $\alpha, \beta \in \mathcal{A}$, $\alpha \,\mathrm{I_R}\, \beta$, if and only if, $\alpha \not\mathrm{R}\, \beta$ and $\beta \not\mathrm{R}\, \alpha$. «

Therefore, if $\alpha \, I_R \, \beta$, then α and β are incomparable with respect to the original relation R.

2.2.2 Connections between Preorder Relations

For the definitions of preorder and associated relations given in Section 2.2 we now take a brief look at some of the relationships between relation R and its associated relations S_R, E_R and I_R. To aid the discussion, first we define R^{-1}, the *inverse* of relation R, as follows.

Definition 2.7 » Inverse of relation R
For some relation R on $\mathcal{A}$, the *inverse relation* R^{-1} of R is given by

— for all $\alpha, \beta \in \mathcal{A}$, $(\beta, \alpha) \in R^{-1}$ if and only if $(\alpha, \beta) \in R$. «

Now we look at Figure 2.1, which shows the basic set relationships between the relations R, E_R, S_R, I_R and R^{-1}. For example, we can see that $R \cap R^{-1} = E_R$, i.e., the intersection of relation R and its inverse relation R^{-1} gives us the equivalence relation E_R. Also we can see that $S_R \cup E_R = R$, i.e., for all $(\alpha, \beta) \in R$, we have either $(\alpha, \beta) \in S_R$, i.e., α is strictly preferred to β, or we have that $(\alpha, \beta) \in E_R$, i.e., α is equivalent to β.

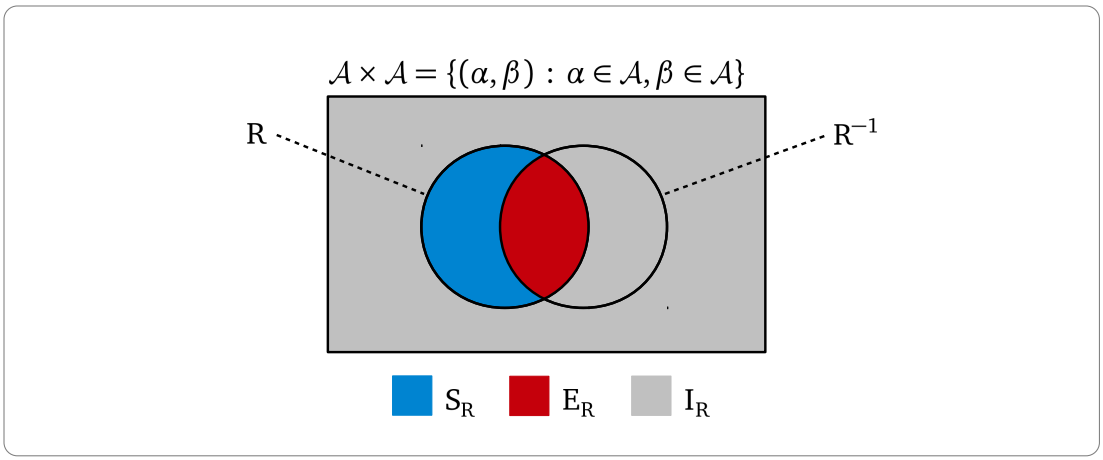


Figure 2.1: The relationships between relations R, S_R, E_R and I_R

We give some further properties of relation R in the following remark.

Remark 2.2 » Preorders and associated relations
For a preorder R on a set $\mathcal{A}$,

(i) If R is antisymmetric, then $E_R = \emptyset$.

(ii) If R is complete, then $I_R = \emptyset$; inversely, if R is not complete, $I_R \neq \emptyset$.

Part (i) of Remark 2.2 gives us that if R is antisymmetric (a partial order or total order), then we have no two distinct elements that are equally preferred. Part (ii) gives us that if relation R is complete (a total preorder or total order), since we have for all $\alpha, \beta \in \mathcal{A}$, either $\alpha \, R \, \beta$, or $\beta \, R \, \alpha$ (or both), then we have no two distinct elements that are incomparable. We also have that if R is not complete then there exists $\alpha, \beta \in \mathcal{A}$ such that α and β are incomparable.

2.2.3 Preferable Elements

Here we introduce some general terms for elements of a given set that are of interest when given a particular preference relation, the elements of $\mathcal{A}$ that are in some way *preferable*. For a preorder R on a set $\mathcal{A}$, with associated relations S_R, E_R and I_R, we consider the following elements of $\mathcal{A}$.

Definition 2.8 » Best element
A element $\alpha \in \mathcal{A}$ is a *best* element for R of $\mathcal{A}$, if and only if

— for all $\beta \in \mathcal{A}$, $\alpha \, R \, \beta$. «

That is, an element α is a best element of $\mathcal{A}$ if it is preferred to all other elements in $\mathcal{A}$. For a preorder R on a set $\mathcal{A}$, let B_R denote the set of best elements of $\mathcal{A}$, with respect to R. We also call this set the *best set*.

Definition 2.9 » Optimal element
An element $\alpha \in \mathcal{A}$ is a *optimal* (or *non-dominated*) element for R of $\mathcal{A}$, if and only if

— there is no element $\beta \in \mathcal{A}$ such that $\beta \, S_R \, \alpha$. «

That is, an element α is an optimal element of $\mathcal{A}$ if there is no other element in $\mathcal{A}$ that is strictly preferred to α. For a preorder R on a set $\mathcal{A}$, let O_{S_R} denote the set of optimal elements of $\mathcal{A}$ with respect to S_R, the strict part of R. We call this the *optimal set*.

2.2.4 Relationship between Optimal and Best Elements

Now we take a look at the relationship between the optimal and best elements of a set $\mathcal{A}$ given some preorder R. Firstly, we have the following remark.

Remark 2.3 » Best and optimal sets
For a preorder R on a set $\mathcal{A}$, we have

(i) $B_R \subseteq O_{S_R}$

(ii) $|O_{S_R}| \geq 1$

(iii) If R is complete, then $|B_R| \geq 1$ and $O_{S_R} = B_R$.

(iv) If R is antisymmetric, then $|B_R| = 0$ or 1.

Part (i) gives us that the best elements are also optimal. Part (ii) gives us that we necessarily have an optimal element. Part (iii) gives us that if R is complete, then we necessarily have a best element and the best and optimal sets coincide. Part (iv) gives us that if R is antisymmetric and there is a best element, it is necessarily unique.

Let us look at an example showing the relationships between the best and optimal sets for different relations.

Example 2.1 ► Best and optimal sets example**.**
Figure 2.2 shows three different example relations R *on some set* $\mathcal{A}$*, where the relation* $\rightarrow$ *is defined by,*

- $[\alpha]_R \rightarrow [\beta]_R$ *if and only if for all* $\alpha \in [\alpha]_R$*, for all* $\beta \in [\beta]_R$*,* $\alpha \, S_R \, \beta$*,*

which is if and only if any element of the equivalence class $[\alpha]_R$ *is strictly preferred to any element of the equivalence class* $[\beta]_R$*.*

Part (a) of Figure 2.2 shows the best and optimal sets for a total preorder R*, where we necessarily have one or more best elements, part (b) of Figure 2.2 shows the optimal set and non-empty best set for a preorder* R*, when* R *is not complete, and part (c) of Figure 2.2 shows the optimal set and empty best set for a preorder* R*, when again* R *is not complete, and the best set is empty.* ▲

We can also see, from part (c) of Figure 2.2, that the optimal set O_{S_R} for relation R on set $\mathcal{A}$ is made up of unions of equivalence classes of E_R.

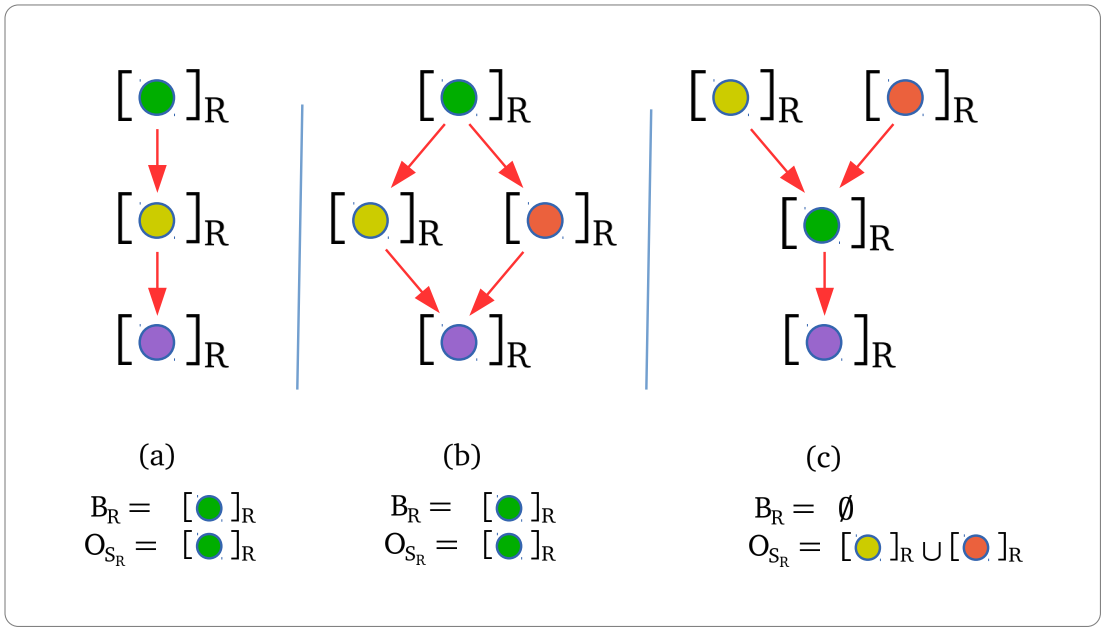


Figure 2.2: The best set B_R and optimal set O_{S_R} for relation R on set $\mathcal{A}$, (a) when R is a total preorder (b) when R is not complete and the best set is not empty, and (c) when R is not complete and there are no best elements.

2.2.5 Extending Relations

To aid the later comparisons of different preference relations, i.e., different R, we define the notion of an *extension* to a relation [FRW10].

Definition 2.10 » Relation extension
An *extension* R' to a relation R on a set $\mathcal{A}$, is a binary relation on $\mathcal{A}$ such that

— for all $\alpha, \beta \in \mathcal{A}$, $\alpha \, R \, \beta \Rightarrow \alpha \, R' \, \beta$ «

So if α is preferred to β according to the original relation, then it is still preferred according to the extending relation. However the extending relation is more discriminating than the original relation as it can compare more pairs of decisions than the original relation. In terms of relations R and R' as sets of ordered pairs, we have that $R \subseteq R'$.

Now we look at the relationships between the optimal sets of preorder relations R and R' on $\mathcal{A}$, where $S_{R'}$, the strict part of R', is an extension of S_R, the strict part of R'. Let O_{S_R} denotes the optimal set of $\mathcal{A}$ with respect to S_R and let $O_{S_{R'}}$ denotes the optimal set of $\mathcal{A}$ with respect to $S_{R'}$. The following result gives us that if α is an optimal element of $\mathcal{A}$ with respect to R', then it is an optimal element of $\mathcal{A}$ with respect to R.

Proposition 2.1 » Optimal sets for preorder and extension

For preorders R, R′ *on a set* $\mathcal{A}$*, such that* $S_{R'}$ *is an extension of* S_R*, i.e., such that* $\alpha\, S_R\, \beta \Rightarrow \alpha\, S_{R'}\, \beta$*, we have that*

— $O_{S_R} \supseteq O_{S_{R'}}$ ◇

Proof: Suppose there exists $\alpha \in \mathcal{A}$ such that $\alpha \notin O_{S_R}$. Then by definition of O_{S_R}, there exists some $\beta \in \mathcal{A}$ such that $\beta\, S_R\, \alpha$. Since $S_{R'}$ extends S_R, then we have $\beta\, S_R\, \alpha$ implies $\beta\, S_{R'}\, \alpha$. This implies $\alpha \notin O_{S_{R'}}$, proving the result. ∎

2.3 Preference Relations and Decision Making

In this section, we look at preference relations in the context of a general decision making problem, where we have a set $\mathcal{A}$ of decisions, alternatives or choices, and depending on the situation the task is to choose a single decision or a subset of decisions from this set, given some preference information relating to each decision. Firstly, we consider a general decision making problem, which is not specific to any particular decision-making field or area, where the purpose of this definition is to facilitate the discussion of different preference relations.

Definition 2.11 » Multi-aspect decision problem
A *multi-aspect decision problem* is a tuple $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where:

- $\mathcal{A}$ is a finite set of decisions, alternatives or choices,
- $\mathcal{S} = \{1, \ldots, m\}$ is a finite set of decision *aspects*, where each $i \in \mathcal{S}$ labels some preferential aspect of the problem, and for which p_i is a function that specifies the preference value of each decision, i.e., $p_i : \mathcal{A} \to T$,
- T is a scale of preference values, where $\leq$ is a total order on T. «

In this definition, each $i \in \mathcal{S}$ is a labelling of some aspect of the problem for which there is a preference value specified, for example, it could be a *criterion* in multi-criteria decision making and $p_i(\alpha)$ is a rating of α in criterion i, or it could be a *state* of the world in decision making under uncertainty, and $p_i(\alpha)$ is the preference level of α should state i occur. To define some compact notation, we let α_i denote

the preference value for decision $\alpha \in \mathcal{A}$ in aspect $i \in \mathcal{S}$, i.e., $\alpha_i = p_i(\alpha)$. For each decision $\alpha \in \mathcal{A}$, we have m preference values $\alpha_1 \ldots, \alpha_m$, one value for each of the given aspects.

In the definition given, we consider a single scale T, however it could be the case that each labelled aspect has a scale of preference values of its own, for example, criteria with different scales. We also consider the *polarity* of the scale, where T could represent positive preferences or different levels of positive outcomes such as utilities or degrees of satisfaction, or T could represent negative preferences, where the values represent different levels of negative outcomes such as costs or degrees of violation. The polarity of the scale T determines whether or not larger or smaller values are preferred according to some preference relation; intuitively, for positive preferences, larger values are preferred, and for negative preferences smaller values are preferred.

Another consideration in these problems is that we can also have different types of scale, for example, T can be *quantitative*, where the difference between two preference degrees has some meaning, or, if T is purely *qualitative*, then we just have an ordering between preference degrees.

2.3.1 Preference Level Representation and Notation

Given a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, we present a couple of different ways of representing the overall preference level of a decision α, i.e, the m preference values $\alpha_1 \ldots, \alpha_m$ associated to α.

Definition 2.12 » Preference vector
The *preference vector* $v(\alpha)$ of a decision $\alpha \in \mathcal{A}$ is given by

— $v(\alpha) = (\alpha_1, \ldots, \alpha_m)$. «

That is, it is the vector of m preference values of decision α given in the order of the decision aspects $1, \ldots, m$. The preference vector representation is important when the ordering of the preference values needs to be maintained, for example, in multi-scale decision making problems such as in multi-criteria decision making [KR76], where we require to compare the preference values of two decisions on a given criterion.

Definition 2.13 » Sorted preference vector
The *sorted preference vector* $\upsilon(\alpha)^{\uparrow}$ of a decision $\alpha \in \mathcal{A}$ is given by

— $\upsilon(\alpha)^{\uparrow} = (\alpha_{(1)}, \ldots, \alpha_{(m)})$, such that, $\alpha_{(1)} \leq \ldots \leq \alpha_{(m)}$. «

That is, $\upsilon(\alpha)^{\uparrow}$ is the unique permutation of $\upsilon(\alpha)$ such that the preference values are reordered in non-decreasing order. The sorted preference vector representations can be used when the ordering of the preference values is not important and the comparison of two decisions does not rely on maintaining the aspect ordering, for example, in social welfare theory [Sen70], where in a social welfare distribution there is no ordering over the individuals.

We use the following notation for the component-wise comparison of two preference vectors, where we have the relations $\leq$, $<$ and $=$, defined over preference vectors of equal size.

Definition 2.14 » Preference vector dominance
For any two preference vectors $\upsilon(\alpha)$ and $\upsilon(\beta)$, of size m, we have that

(i) $\upsilon(\alpha) \leq \upsilon(\beta)$ if and only if $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$,

(ii) $\upsilon(\alpha) < \upsilon(\beta)$ if and only if $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$, and there exists $j \in \{i, \ldots, m\}$ such that $\alpha_j < \beta_j$.

(iii) $\upsilon(\alpha) = \upsilon(\beta)$ if and only if $\alpha_i = \beta_i$ for all $i \in \{1, \ldots, m\}$. «

2.4 Pareto Dominance

In this section, we look at a preference relation called Pareto dominance. This is with a view to providing some background on the Sorted-Pareto dominance relation (see Section 2.5), which is an extension to Pareto dominance and is one of the main focuses of the thesis.

The Pareto dominance relation [Par97] prefers decisions that are at least as good in every aspect, and strictly better in at least one aspect [Sen70, Ch. 2]. For example, in multi-criteria decision making, if we are comparing two decisions, then Pareto dominance will prefer the decision that is at least as good in every criteria and strictly better in at least one. Here we give definitions for the Pareto dominance relations, where we assume some multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$,

for which T is a negative preference scale, i.e., we are minimising and smaller values are preferred.

First we give a definition for the Weak Pareto dominance relation, which is a preorder on $\mathcal{A}$ (see Definition 2.2).

Definition 2.15 » Weak Pareto dominance
For all $\alpha, \beta \in \mathcal{A}$, α *Weak Pareto dominates* β, written as $\alpha \preccurlyeq_{\mathrm{P}} \beta$, if and only if

— $v(\alpha) \leq v(\beta)$ «

That is, we have that $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$. For example, for two decisions α and β, with preference vectors $v(\alpha) = (2, 3)$ and $v(\beta) = (2, 4)$, we can see that $\alpha \preccurlyeq_{\mathrm{P}} \beta$, since α is at least as good as β for all $i \in \{1, \ldots, m\}$. The strict version of Pareto dominance is defined as follows.

Definition 2.16 » Pareto dominance
For all $\alpha, \beta \in \mathcal{A}$, α *Pareto dominates* β, written as $\alpha \prec_{\mathrm{P}} \beta$, if and only if

— $v(\alpha) < v(\beta)$ «

That is, we have $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$, and there exists $j \in \{i, \ldots, m\}$ such that $\alpha_j < \beta_j$. For example, for two decisions α and β, with preference vectors $v(\alpha) = (2, 3)$ and $v(\beta) = (2, 4)$, we can see that $\alpha \prec_{\mathrm{P}} \beta$, since α is at least as good as β for all $i \in \{1, \ldots, m\}$, and strictly better for $j = 2$.

Since Pareto dominance is the strict or asymmetric part of weak Pareto dominance (see Definition 2.3), then $\prec_{\mathrm{P}}$ can also be defined in terms of $\preccurlyeq_{\mathrm{P}}$, which we give in the following remark.

Remark 2.4 » Pareto dominance in terms of $\preccurlyeq_{\mathrm{P}}$
For all $\alpha, \beta \in \mathcal{A}$, $\alpha \prec_{\mathrm{P}} \beta$ if and only if $\alpha \preccurlyeq_{\mathrm{P}} \beta$ and $\beta \not\preccurlyeq_{\mathrm{P}} \alpha$.

Now we look at the equivalence relation for Pareto dominance, which is defined as follows.

Definition 2.17 » Pareto equivalence
For all $\alpha, \beta \in \mathcal{A}$, α is *Pareto equivalent* β, written as $\alpha \equiv_{\mathrm{P}} \beta$, if and only if

— $v(\alpha) = v(\beta)$ «

That is, we have that $\alpha_i = \beta_i$ for all $i \in \{1, \ldots, m\}$. For example, for two decisions γ and δ, with preference vectors $v(\gamma) = (4, 2)$ and $v(\delta) = (4, 2)$, we can see that $\gamma \equiv_{\mathrm{P}} \delta$, since $\gamma_i = \delta_i$, for all $i \in \{1, \ldots, m\}$.

Since Pareto equivalence is the symmetric part of weak Pareto dominance (see Definition 2.4), then $\equiv_{\mathrm{P}}$ can also be defined in terms of $\preccurlyeq_{\mathrm{P}}$, which we give in the following remark.

Remark 2.5 » Pareto equivalence in terms of $\preccurlyeq_{\mathrm{P}}$
For all $\alpha, \beta \in \mathcal{A}$, $\alpha \equiv_{\mathrm{P}} \beta$ if and only if $\alpha \preccurlyeq_{\mathrm{P}} \beta$ and $\beta \preccurlyeq_{\mathrm{P}} \alpha$.

We now look at the notion of non-dominated or optimal decisions when considering the Pareto dominance relation. In the context of Definition 2.9, these decisions are the optimal elements of $\mathcal{A}$ with respect to the $\prec_{\mathrm{P}}$ relation. It is a desirable property in decision making, since for any decision in $\mathcal{A}$ that is not Pareto optimal, there exists some other decision in $\mathcal{A}$ that is at least as good in every aspect and strictly better in some.

Definition 2.18 » Pareto non-dominated/optimal decision
A decision $\alpha \in \mathcal{A}$ is *Pareto non-dominated* (or Pareto optimal) if and only if

— there is no $\beta \in \mathcal{A}$ such that $\beta \prec_{\mathrm{P}} \alpha$. «

That is, if and only if it is not Pareto dominated by any other decision. We denote the set of Pareto non-dominated decisions of $\mathcal{A}$ as O_{P}. Let us look at an example:

Example 2.2 ► Pareto optimal example**.**
Consider some multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where

- $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$
- $\mathcal{S} = \{1, 2\}$
- $T = \{low, med, hi\}$

where T is a scale of cost estimates, ordered by $\leq$.

The preference vector for each decision is given as follows.

- $v(\alpha) = (low, med)$
- $v(\beta) = (low, hi)$
- $v(\gamma) = (hi, low)$

- $v(\delta) = (hi, low)$

Since we have $\alpha \prec_{\mathrm{P}} \beta$, then β is not Pareto optimal, i.e., we have that

- $\beta \notin \mathrm{O_P}$

However, there exists no decision in $\mathcal{A}$ that Pareto-dominates α, γ or δ, so we have that

- $\mathrm{O_P} = \{\alpha, \gamma, \delta\}$ ▲

2.4.1 Pareto Dominance on Preference Vectors

The interpretation of Pareto dominance given in Section 2.4 is as a relation on a set of decisions $\mathcal{A}$ in a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where for each decision $\alpha \in \mathcal{A}$ we have a preference vector $v(\alpha)$ of m values from a given scale T. We can see that the preference vector space is T^m, the set of *m-tuples* of T, which is $T^m = \underbrace{T \times \cdots \times T}_{m}$.

If instead we consider Pareto dominance as a relation on the preference vector space T^m, then using preference vector dominance (see Definition 2.14), we induce the Pareto dominance relations over T^m, as follows.

Remark 2.6 » Pareto dominance on T^m
For all $\mathbf{u}, \mathbf{v} \in T^m$, we have

- $\mathbf{u} \preccurlyeq_{\mathrm{P}} \mathbf{v}$ if and only if $\mathbf{u} \leq \mathbf{v}$,
- $\mathbf{u} \prec_{\mathrm{P}} \mathbf{v}$ if and only if $\mathbf{u} < \mathbf{v}$,
- $\mathbf{u} \equiv_{\mathrm{P}} \mathbf{v}$ if and only if $\mathbf{u} = \mathbf{v}$.

2.5 Sorted-Pareto Dominance

As mentioned in the previous section, Pareto optimality is a desirable property when choosing decisions in a multi-aspect decision problem, however the Pareto dominance relation is not very discerning since many comparisons between pairs of decisions do not result in dominance. In this section, we look at Sorted-Pareto dominance, which is also called *Ordered Pareto* [KP08], or *Symmetric Pareto* [DPT13], and it can be used in situations when the ordering of the decision aspects is not important, or when the scales used in each aspect can be made commensurate. Since it is more discriminating than Pareto dominance, it produces a much smaller

optimal set of decisions, and since the Sorted-Pareto dominance extends the Pareto dominance relation (see Proposition 2.4), then Sorted-Pareto optimal decisions are also Pareto optimal.

Here we define the Sorted-Pareto dominance relations, assuming some multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where we have a negative preference scale T. First we give the definition for Weak Sorted-Pareto dominance, which is a preorder on $\mathcal{A}$.

Definition 2.19 » Weak Sorted-Pareto dominance
For all $\alpha, \beta \in \mathcal{A}$, α *Weak Sorted-Pareto dominates* β, written as $\alpha \preccurlyeq_{\text{SP}} \beta$, iff.

— $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$ «

That is, if and only if $\alpha_{(i)} \leq \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$. The strict version of the relation is defined as follows.

Definition 2.20 » Sorted-Pareto dominance
For all $\alpha, \beta \in \mathcal{A}$, α *Sorted-Pareto dominates* β, written as $\alpha \prec_{\text{SP}} \beta$, if and only if

— $v(\alpha)^{\uparrow} < v(\beta)^{\uparrow}$ «

That is, if and only if $\alpha_{(i)} \leq \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$, and there exists $j \in \{1, \ldots, m\}$ such that $\alpha_{(j)} < \beta_{(j)}$. From Definition 2.3, the Sorted-Pareto dominance relation $\prec_{\text{SP}}$ can also be defined in terms of $\preccurlyeq_{\text{SP}}$, which we give in the following result.

Proposition 2.2 » Sorted-Pareto dominance in terms of $\preccurlyeq_{\text{SP}}$

For all $\alpha, \beta \in \mathcal{A}$, $\alpha \prec_{\text{SP}} \beta$ if and only if

— $\alpha \preccurlyeq_{\text{SP}} \beta$ *and* $\beta \not\preccurlyeq_{\text{SP}} \alpha$. ◇

Proof: Suppose $\alpha \prec_{\text{SP}} \beta$, Then by definition of $\prec_{\text{SP}}$, we have $\alpha_{(i)} \leq \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$, which means by definition of $\preccurlyeq_{\text{SP}}$ we have that $\alpha \preccurlyeq_{\text{SP}} \beta$. By definition of $\prec_{\text{SP}}$, we also have that there exists $j \in \{1, \ldots, m\}$ such that $\alpha_{(j)} < \beta_{(j)}$, which means by definition of $\preccurlyeq_{\text{SP}}$, we cannot have that $\beta \preccurlyeq_{\text{SP}} \alpha$.

Now we prove the converse. Suppose $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \not\preccurlyeq_{\text{SP}} \alpha$. By definition of $\preccurlyeq_{\text{SP}}$, we have $\alpha_{(i)} \leq \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$. Since $\beta \not\preccurlyeq_{\text{SP}} \alpha$, then it is not the case that $\beta_{(i)} \leq \alpha_{(i)}$ for all $i \in \{1, \ldots, m\}$, i.e., there exists some $j \in \{1, \ldots, m\}$ such that $\beta_{(j)} \not\leq \alpha_{(j)}$, i.e., $\alpha_{(j)} < \beta_{(j)}$. By definition of $\prec_{\text{SP}}$, this gives us that $\alpha \prec_{\text{SP}} \beta$. ∎

The equivalence relation for Sorted-Pareto is defined as follows.

Definition 2.21 » Sorted-Pareto equivalence
For all $\alpha, \beta \in \mathcal{A}$, α is *Sorted-Pareto equivalent* to β, written as $\alpha \equiv_{\text{SP}} \beta$, if and only if

— $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$ «

That is, if and only if $\alpha_{(i)} = \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$, i.e., $v(\alpha)$ and $v(\beta)$ are permutations of each other. Referring to Definition 2.4, we can see that Sorted-Pareto equivalence can also be defined in terms of Weak Sorted-Pareto dominance. We have the following result.

Proposition 2.3 » Sorted-Pareto equivalence in terms of $\preccurlyeq_{\text{SP}}$

For all $\alpha, \beta \in \mathcal{A}$, $\alpha \equiv_{\text{SP}} \beta$ if and only if

— $\alpha \preccurlyeq_{\text{SP}} \beta$ *and* $\beta \preccurlyeq_{\text{SP}} \alpha$. ◇

Proof: Suppose $\alpha \equiv_{\text{SP}} \beta$. By definition of $\equiv_{\text{SP}}$, we have $\alpha_{(i)} = \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$. It follows from the definition of $\preccurlyeq_{\text{SP}}$ that $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \preccurlyeq_{\text{SP}} \alpha$.

Now suppose $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \preccurlyeq_{\text{SP}} \alpha$. By definition of $\preccurlyeq_{\text{SP}}$ we have, for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$ and $\beta_{(i)} \leq \alpha_{(i)}$, i.e., for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} = \beta_{(i)}$, which gives us, by definition of $\equiv_{\text{SP}}$, that $\alpha \equiv_{\text{SP}} \beta$. ▪

Given the Sorted-Pareto equivalence definition, we also have the notion of a Sorted-Pareto *equivalence class* for a decision, which we define as follows.

Definition 2.22 » Sorted-Pareto equivalence class
The *Sorted-Pareto equivalence class* of $\alpha \in \mathcal{A}$, denoted by $[\alpha]_{\text{SP}}$, is defined as

— $[\alpha]_{\text{SP}} = \{\beta : \beta \in \mathcal{A}, \beta \equiv_{\text{SP}} \alpha\}$ «

Therefore for some $\alpha \in \mathcal{A}$, the set $[\alpha]_{\text{SP}}$ is the set all the decisions that are Sorted-Pareto equivalent to α.

For a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, the optimal decisions of $\mathcal{A}$ with respect to the $\prec_{\text{SP}}$ relation (see Definition 2.9) are called *Sorted-Pareto non-dominated* decisions, which are defined as follows.

Definition 2.23 » Sorted-Pareto non-dominated/optimal decision
A decision $\alpha \in \mathcal{A}$ is *Sorted-Pareto non-dominated/optimal*, if and only if

— there is no $\beta \in \mathcal{A}$ such that $\beta \prec_{\text{SP}} \alpha$. «

That is, if and only if it is not Sorted-Pareto dominated by any other decision. We denote the set of Sorted-Pareto non-dominated decisions as O_{SP}.

Now we look at an example which compares the resulting Pareto optimal and Sorted-Pareto optimal sets of decisions for a given problem.

Example 2.3 ► Pareto and Sorted-Pareto optimal example**.**
Consider again the multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$ from Example 2.2, where we have:

- $v(\alpha) = (low, med)$
- $v(\beta) = (low, hi)$
- $v(\gamma) = (hi, low)$
- $v(\delta) = (hi, low)$

The Pareto optimal set of $\mathcal{A}$ is given by:

- $\text{O}_{\text{P}} = \{\alpha, \gamma, \delta\}$.

Now, if we use Sorted-Pareto dominance instead of Pareto dominance, then the sorted preference vectors of the decisions are given by:

- $v(\alpha) = (low, med)$
- $v(\beta) = (low, hi)$
- $v(\gamma) = (low, hi)$
- $v(\delta) = (low, hi)$

We can see that $\alpha \prec_{\text{SP}} \beta$, $\alpha \prec_{\text{SP}} \gamma$ and $\alpha \prec_{\text{SP}} \delta$, and there is no decision in $\mathcal{A}$ that Sorted-Pareto dominates α, so we have that:

- $\text{O}_{\text{SP}} = \{\alpha\}$

Now there is a smaller set of optimal decisions from which to choose; in this instance O_{SP} is a strict subset of O_{P}. Since α is now the only optimal decision with respect to the

Sorted-Pareto relation, there is an argument for a decision maker to choose decision α over any of the others. ▲

As already illustrated in Example 2.3, the Sorted-Pareto dominance relation extends the Pareto dominance relation, and given Proposition 2.1, we have that O_{SP} is a subset of O_P. We give the following result.

Proposition 2.4 » Sorted-Pareto extends Pareto result

For a multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, we have that:

(i) $\preccurlyeq_{SP}$ *extends* $\preccurlyeq_P$ *i.e., for all* $\alpha, \beta \in \mathcal{A}$*, if* $\alpha \preccurlyeq_P \beta$ *then* $\alpha \preccurlyeq_{SP} \beta$*.*

(ii) $\prec_{SP}$ *extends* $\prec_P$*, i.e., for all* $\alpha, \beta \in \mathcal{A}$*, if* $\alpha \prec_P \beta$*, then* $\alpha \prec_{SP} \beta$*.* ◇

Proof:

(i) Suppose $\alpha \preccurlyeq_P \beta$. Therefore, by definition of $\preccurlyeq_P$, $\alpha_i \leq \beta_i$, for all $i = 1, \ldots, m$. Consider any $j \in \{1, \ldots, m\}$. $\beta_{(j)}$ is the j-th smallest element of β, so there are at least j components β_i of β with $\beta_i \leq \beta_{(j)}$. If $\beta_i \leq \beta_{(j)}$, then $\alpha_i \leq \beta_{(j)}$, since $\alpha_i \leq \beta_i$ by definition of $\preccurlyeq_P$. Therefore, there are at least j components α_i of α with $\alpha_i \leq \beta_{(j)}$, in particular $\alpha_{(j)} \leq \beta_{(j)}$. Therefore, $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$, which by definition of $\preccurlyeq_{SP}$, implies that $\alpha \preccurlyeq_{SP} \beta$.

(ii) Suppose $\alpha \prec_P \beta$. Then we have, by definition of $\prec_P$, that $\alpha \preccurlyeq_P \beta$, and therefore from (i) we have that $\alpha \preccurlyeq_{SP} \beta$, which means we must have either $\alpha \equiv_{SP} \beta$ or $\alpha \prec_{SP} \beta$. Suppose $\alpha \equiv_{SP} \beta$, and we prove a contradiction here. By definition of $\prec_P$, we have that there exists some $j \in \{1, \ldots, m\}$ such that $\alpha_j < \beta_j$. Let $j_\alpha \in \{1, \ldots, m\}$ be such that $\alpha_{(j_\alpha)} = \alpha_j$, and let $j_\beta \in \{1, \ldots, m\}$ be such that $\beta_{(j_\beta)} = \beta_j$. Then since we have from our supposition $\alpha_{(i)} = \beta_{(i)}$ for all $i \in \{1, \ldots, m\}$, then we must have that $j_\alpha < j_\beta$. Therefore there must exists some other $k \in \{1, \ldots, m\}$ such that $\beta_k > \alpha_k$. However this contradicts $\alpha \prec_P \beta$, so therefore we must have that $\alpha \prec_{SP} \beta$. ▪

2.5.1 Sorted-Pareto Dominance on Preference Vectors

The definitions of the Sorted-Pareto dominance relations given in the previous section are defined over the set of decisions $\mathcal{A}$ in a multi-aspect decision problem

$P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$. Similar to as done with Pareto dominance in Section 2.4.1, if we let the set of decisions $\mathcal{A}$ be equal to the preference vector space $T^m = \underbrace{T \times \cdots \times T}_{m}$, then we induce the Sorted-Pareto dominance relations over T^m, again using preference vector dominance from Definition 2.14.

Remark 2.7 » Sorted-Pareto dominance on T^m
For all $\mathbf{u}, \mathbf{v} \in T^m$, we have

- $\mathbf{u} \preccurlyeq_{\text{SP}} \mathbf{v}$ if and only if $\mathbf{u}^{\uparrow} \leq \mathbf{v}^{\uparrow}$,
- $\mathbf{u} \prec_{\text{SP}} \mathbf{v}$ if and only if $\mathbf{u}^{\uparrow} < \mathbf{v}^{\uparrow}$,
- $\mathbf{u} \equiv_{\text{SP}} \mathbf{v}$ if and only if $\mathbf{u}^{\uparrow} = \mathbf{v}^{\uparrow}$.

2.6 Connections between Sorted-Pareto Dominance and Other Relations

In this section, we look at some other preference relations that are connected to Sorted-Pareto dominance. Since the Sorted-Pareto relation involves comparing decisions based on a reordering of the preference vectors associated with the decisions, then we look at other preference relations that also use such a reordering. The definition of Sorted-Pareto dominance given in the previous section is in the context of minimising negative preferences, for example, *costs*, so smaller values on our scale T are preferred. We look at some preference relations that are related to Sorted-Pareto in the context of minimising, but we also examine the connection between Sorted-Pareto and relations in the context of maximising positive preferences. We also discuss relations that use on additional information such as quantitative preferential information or probability information on scenarios.

In the positive preference scale case, the direction of *preference* is reversed from the negative preference scale case, so we have that if $\alpha \succcurlyeq_{\text{SP}} \beta$, then α is *Weak Sorted-Pareto preferred* to β, and we have that if $\alpha \succ_{\text{SP}} \beta$, then α is *Sorted-Pareto preferred* to β. This also gives us that a decision $\alpha \in \mathcal{A}$ is *Sorted-Pareto optimal* if and only if there is no $\beta \in \mathcal{A}$ such that $\beta \succ_{\text{SP}} \alpha$. To differentiate between the Sorted-Pareto optimal sets for minimising and maximising preferences, we let O_{SP^*} denote the set of Sorted-Pareto optimal decisions for maximising positive preferences.

To aid the comparison between Sorted-Pareto dominance and other preference relations, we first give the following result on extending Sorted-Pareto dominance,

which is a new result.

Lemma 2.1 » Extending Sorted-Pareto result

(i) Let $\leq_R$ be a preorder on T^m that extends Weak-Pareto dominance $\preccurlyeq_{\mathrm{P}}$ defined on T^m. Define relation $\preccurlyeq$ on $\mathcal{A}$ by, $\alpha \preccurlyeq \beta \iff \upsilon(\alpha)^{\uparrow} \leq_R \upsilon(\beta)^{\uparrow}$.

Then $\preccurlyeq$ extends $\preccurlyeq_{\mathrm{SP}}$.

(ii) Let $<_S$ be a preorder on T^m that extends Pareto dominance $\prec_{\mathrm{P}}$ defined on T^m. Define relation $\prec$ on $\mathcal{A}$ by, $\alpha \prec \beta \iff \upsilon(\alpha)^{\uparrow} <_S \upsilon(\beta)^{\uparrow}$.

Then $\prec$ extends $\prec_{\mathrm{SP}}$. ◇

Proof:

(i) Suppose that $\alpha \preccurlyeq_{\mathrm{SP}} \beta$. Then we have by definition of $\preccurlyeq_{\mathrm{SP}}$ that $\upsilon(\alpha)^{\uparrow} \leq \upsilon(\beta)^{\uparrow}$. Since $\leq_R$ extends $\leq$, then we have $\upsilon(\alpha)^{\uparrow} \leq \upsilon(\beta)^{\uparrow} \Rightarrow \upsilon(\alpha)^{\uparrow} \leq_R \upsilon(\beta)^{\uparrow}$, so by definition of $\preccurlyeq$, we have that $\alpha \preccurlyeq \beta$.

(ii) Suppose that $\alpha \prec_{\mathrm{SP}} \beta$. Then have by definition of $\prec_{\mathrm{SP}}$ that $\upsilon(\alpha)^{\uparrow} < \upsilon(\beta)^{\uparrow}$. Since $<_S$ extends $<$, then we have that $\upsilon(\alpha)^{\uparrow} < \upsilon(\beta)^{\uparrow} \Rightarrow \upsilon(\alpha)^{\uparrow} <_S \upsilon(\beta)^{\uparrow}$, and by definition of $\prec$, we have that $\alpha \prec \beta$. ∎

Lemma 2.1 implies that Sorted-Pareto dominance is extended by the Lexicographical Max Ordering [Ehr96] and Leximin [DFP96b, PSS06] total preorders. Sorted-Pareto is also extended by Generalized Lorenz dominance [Sho83], and by Ordered Weighted Averages [Yag88], which are other refinements of Pareto dominance, however unlike these relations Sorted-Pareto can be used for purely ordinal or qualitative information aggregation.

2.6.1 Sorted-Pareto and Minimax

In this section we relate Sorted-Pareto dominance to the Minimax relation, which is a preference relation for minimising negative qualitative preferences. When comparing any two decision in our set of decisions, the Minimax relation [vN28] prefers the decision that has a smaller maximum negative preference level. For example, in game theory, if we are comparing two decisions, and the preference

values indicate possible loss, Minimax will prefer the decision with the smaller maximum loss. We give a definition as follows.

Definition 2.24 » Minimax
For all $\alpha, \beta \in \mathcal{A}$, α is *Minmax-preferred* to β, written as $\alpha \preceq_{\text{MX}} \beta$, if and only if

— $\max v(\alpha) \leq \max v(\beta)$ «

That is, we have that $\alpha \preceq_{\text{MX}} \beta$ if and only if $\alpha_{(m)} \leq \beta_{(m)}$. We also have a strict version of the Minimax relation, where there is a strict preference between two decisions.

Definition 2.25 » Strict Minimax
For all $\alpha, \beta \in \mathcal{A}$, α is *strictly Minimax-preferred* to β, written as $\alpha \prec_{\text{MX}} \beta$, iff

— $\max v(\alpha) < \max v(\beta)$ «

That is, we have that $\alpha \prec_{\text{MX}} \beta$ if and only if $\alpha_{(m)} < \beta_{(m)}$. The equivalence relation for Minimax is defined as follows.

Definition 2.26 » Minimax equivalence
For all $\alpha, \beta \in \mathcal{A}$, α is *Minimax equivalent* to β, written as $\alpha \equiv_{\text{MX}} \beta$, if and only if

— $\max v(\alpha) = \max v(\beta)$ «

That is, we have that $\alpha \equiv_{\text{MX}} \beta$ if and only if $\alpha_{(m)} = \beta_{(m)}$.

From the definitions, we can see that the Minimax relation extends the Weak Sorted-Pareto relation, so we have the following result.

Proposition 2.5 » Sorted-Pareto and Minimax

For all $\alpha, \beta \in \mathcal{A}$, we have

— $\alpha \preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \preceq_{\text{MX}} \beta$. ◇

Proof: This follows from the definitions and from Lemma 2.1 (i). ∎

Let B_{MX} denote the set of best decisions from $\mathcal{A}$ with respect to $\preceq_{\text{MX}}$, and let O_{MX} denote the set of optimal decisions from $\mathcal{A}$ with respect to $\prec_{\text{MX}}$. The Minimax relation $\preceq_{\text{MX}}$ forms a total preorder on $\mathcal{A}$, so we have that $\text{B}_{\text{MX}} = \text{O}_{\text{MX}} \neq \emptyset$.

Let us look at an example.

Example 2.4 ► Minimax example**.**
For some scale $T = \{1, 2, 3, 4, 5\}$, which is ordered by $\leq$, let us consider the preference vectors for two decisions α and β, given by

- $v(\alpha) = (1, 4, 1)$
- $v(\beta) = (4, 4, 4)$

We can see that

- $\max v(\alpha) = 4$
- $\max v(\beta) = 4$

so we have $\alpha \not\succ_{\text{MX}} \beta$, and $\beta \not\succ_{\text{MX}} \alpha$, i.e., neither is strictly preferred to the other with respect to the Minimax relation.

However, we can see that for all preference values other than the maximum value, α is better than β, i.e., we have that

- $\alpha_{(1)} < \beta_{(1)}$
- $\alpha_{(2)} < \beta_{(2)}$

but these comparisons are not considered by Minimax. We can see however that α Pareto dominates β. ▲

This example shows what is known as the "Drowning effect" [DF05, FLS93], since the other preference values in the preference vector besides the maximum value are not considered by the relation. The Sorted-Pareto preference relation does not suffer from this since it considers all values in the preference vector in its comparison.

2.6.2 Sorted-Pareto and Lexicographic-Max

In this section, we compare Sorted-Pareto dominance with the Lexicographic-Max ordering, another preference relation for minimising negative preferences. The Lexicographic-Max ordering [Ehr96] is similar to Minimax in that when it compares two decisions, it prefers the one with the smaller maximum preference value, however if both maxima are equal then it will lexicographically compare the next largest preference value of each decision until it finds that they are not equal. Therefore the

Lexicographic-Max ordering is an extension of MinMax. First we define the strict Lexicographic-Max ordering $\prec_{\text{LMX}}$ as follows.

Definition 2.27 » Strict Lexicographic-Max
For all $\alpha, \beta \in \mathcal{A}$, α is *strictly Lexicographic-Max preferred* to β, written as $\alpha \prec_{\text{LMX}} \beta$, if and only if there exists some $j \in \{1, \ldots, m\}$ such that

(a) for all $i \in \{j+1, \ldots, m\}$, $\alpha_{(i)} = \beta_{(i)}$, and

(b) $\alpha_{(j)} < \beta_{(j)}$ «

Now we define the Lexicographic-Max equivalence ordering $\equiv_{\text{LMX}}$.

Definition 2.28 » Lexicographic-Max equivalence
For all $\alpha, \beta \in \mathcal{A}$, α is *Lexicographic-Max equivalent* to β, written as $\alpha \equiv_{\text{LMX}} \beta$, iff

— $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$ «

From this, we can define the Lexicographic-Max ordering $\preceq_{\text{LMX}}$ as follows, in terms of $\prec_{\text{LMX}}$ and $\equiv_{\text{LMX}}$.

Definition 2.29 » Lexicographic-Max
For all $\alpha, \beta \in \mathcal{A}$, α is *Lexicographic-Max preferred* to β, written as $\alpha \preceq_{\text{LMX}} \beta$, iff

— $\alpha \equiv_{\text{LMX}} \beta$ or $\alpha \prec_{\text{LMX}} \beta$. «

Let B_{LMX} denote the set of best decisions from $\mathcal{A}$ with respect to $\preceq_{\text{LMX}}$, and let O_{LMX} denote the set of optimal decisions from $\mathcal{A}$ with respect to $\prec_{\text{LMX}}$. The Lexicographic-Max ordering relation $\preceq_{\text{LMX}}$ forms a total preorder on $\mathcal{A}$, so we have that $\text{B}_{\text{LMX}} = \text{O}_{\text{LMX}} \neq \emptyset$. Comparing the Sorted-Pareto dominance relation and Lexicographic-Max ordering, we can see that the Sorted-Pareto relations are extended by Lexicographic-Max relations, so we have the following result.

Proposition 2.6 » Sorted-Pareto and Lexicographic-Max

For a set of decisions $\mathcal{A}$, where $\alpha, \beta \in \mathcal{A}$, we have:

(i) $\alpha \preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \preceq_{\text{LMX}} \beta$

(ii) $\alpha \prec_{\text{SP}} \beta \Rightarrow \alpha \prec_{\text{LMX}} \beta$

(iii) $\mathrm{O}_{\text{SP}} \supseteq \mathrm{O}_{\text{LMX}}$ ◇

Proof:

(i) This follows from the definitions and from Lemma 2.1 (i).

(ii) This follows from the definitions and from Lemma 2.1 (ii).

(iii) This follows from (ii) and from Proposition 2.1. ∎

Now, let us look at an example using the Lexicographic-Max relation.

Example 2.5 ► Lexicographic-Max example**.**
For some multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, let us consider the preference vectors for two decisions α and β, given by:

- $v(\alpha) = (8, 10, 8, 7)$
- $v(\beta) = (1, 10, 9, 1)$

The sorted preference vectors are given by:

- $v(\alpha)^{\uparrow} = (7, 8, 8, 10)$
- $v(\beta)^{\uparrow} = (1, 1, 9, 10)$

We have that $\alpha \not\preccurlyeq_{\text{SP}} \beta$ and $\beta \not\preccurlyeq_{\text{SP}} \alpha$, therefore we have:

- $\mathrm{O}_{\text{SP}} = \{\alpha, \beta\}$

We also have that $\alpha \prec_{\text{LMX}} \beta$, since $\alpha_{(4)} = \beta_{(4)}$ and $\alpha_{(3)} < \beta_{(3)}$, so we have:

- $\mathrm{O}_{\text{LMX}} = \{\alpha\}$

and we can see that in this example $\mathrm{O}_{\text{SP}} \supset \mathrm{O}_{\text{LMX}}$.

However we also have that decision β is much better than decision α for the remaining components in the preference vectors, i.e.,

- $\beta_{(2)} < \alpha_{(2)}$, *where* $1 < 8$, *and*

- $\beta_{(1)} < \alpha_{(1)}$, *where* $1 < 7$

but these comparisons are not considered by the Lexicographic-Max ordering. ▲

The example here illustrates that even though the Lexicographic-Max ordering does not suffer as much from the drowning effect as the Minimax relation, it can still effectively "ignore" preference values when comparing decisions.

2.6.3 Sorted-Pareto and Maximin

The Maximin relation [Raw71, Wal50] is the maximising counterpart to the Minimax relation already defined. When comparing two decisions the Maximin relation, prefers the decision that has a larger minimum positive preference level. For example, if each value in the preference vector represents some income or utility gained by a person, then the Maximin relation will prefer the decision that has the larger minimum value, therefore maximising the utility of the worst off person. It is defined as follows.

Definition 2.30 » Maximin
For all $\alpha, \beta \in \mathcal{A}$, α is *Maximin-preferred* to β, written as $\alpha \succeq_{\text{MN}} \beta$, if and only if

— $\min v(\alpha) \geq \min v(\beta)$ «

Similar to Minimax, for Maximin we can also define the strict relation $\succ_{\text{MN}}$ and the equivalence relation $\equiv_{\text{MN}}$ in the obvious ways, where $\alpha \succ_{\text{MN}} \beta$ if and only if $\min v(\alpha) > \min v(\beta)$, and $\alpha \equiv_{\text{MN}} \beta$ if and only if $\min v(\alpha) = \min v(\beta)$. As with the Minimax relation, Maximin is a total preorder on the set of decisions $\mathcal{A}$, and we have that the Weak Sorted-Pareto dominance relation is extended by Maximin, which can be shown in a similar way to Proposition 2.5.

2.6.4 Sorted-Pareto and Leximin

The Leximin relation [BJ88, DFP96b, FLS93] is the lexicographical version of the Maximin relation, or in other terms, it is the maximising equivalent of the Lexicographic-Max ordering defined in Section 2.6.2. Leximin is defined as follows.

Definition 2.31 » Strict Leximin
For all $\alpha, \beta \in \mathcal{A}$, α is *strictly Leximin-preferred* to β, written as $\alpha \succ_{\text{LMN}} \beta$, if and only if there exists some $j \in \{1, \ldots, m\}$ such that

(a) for all $i \in \{1, \ldots, j-1\}$, $\alpha_{(i)} = \beta_{(i)}$, and

(b) $\alpha_{(j)} > \beta_{(j)}$ «

Similar to the Lexicographic-Max ordering, for Leximin we have the equivalence relation $\equiv_{\text{LMN}}$, where $\alpha \equiv_{\text{LMN}} \beta$ if and only if $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$, and we have the weak version $\succeq_{\text{LMN}}$, where $\alpha \succeq_{\text{LMN}} \beta$ if and only if $\alpha \succ_{\text{LMN}} \beta$ or $\alpha \equiv_{\text{LMN}} \beta$. Leximin compares two decisions by comparing the minimum preference values in each preference vector with one another, and if these are equal, it proceeds to the next smallest value in the vector. The Leximin ordering is a total preorder on $\mathcal{A}$.

We can perform a similar comparison between Leximin and Sorted-Pareto as is done for Lexicographic-Max ordering and Sorted-Pareto in Proposition 2.6 to show that the Sorted-Pareto is extended by Leximin.

2.6.5 Sorted-Pareto and Sum of Weights

Aside from the information that is available given a multi-aspect decision problem as defined in Definition 2.11, there may be further information available to consider. We could have that the values in T are numerical, or we could have some function $f : T \to \mathbb{R}^+$ that maps our scale T to a numerical scale. In these cases one way of comparing decisions is to perform an aggregation, e.g., summation, of the preference values associated with each decision and then compare the resulting aggregations to see which are preferable.

For example, in weighted constraints (see Section 3.3.2), decisions with a lower sum of negative preference values are preferred, and in *Generalised Additive Independence* (GAI) networks [BG95, BPPS06, GPD11], decisions with a higher sum of positive preference values are preferred. For a function $f : T \to \mathbb{R}^+$ to preserve the order of the scale T, we require that f is *monotonic* with respect to the scale T. We define a monotonic function as follows.

Definition 2.32 » Monotonic function
For some ordered scale T, ordered by $\leq$, a function $f : T \to \mathbb{R}^+$ is *monotonic* with respect to scale T, if and only if

— $u \leq v \Rightarrow f(u) \leq f(v)$, for all $u, v \in T$ «

We could also have that f is *strictly monotonic* with respect to the scale T. We define a strictly monotonic function as follows.

Definition 2.33 » Strictly monotonic function
For some ordered scale T, ordered by $\leq$, a function $f : T \to \mathbb{R}^+$ is *strictly monotonic* with respect to scale T, if and only if

— $u < v \Leftrightarrow f(u) < f(v)$ for all $u, v \in T$

or equivalently, if and only if

— $u \leq v \Leftrightarrow f(u) \leq f(v)$ for all $u, v \in T$ «

Here we define the Min-sum relation for minimising negative preferences.

Definition 2.34 » Min-sum preferred
For all $\alpha, \beta \in \mathcal{A}$, for some function $f : T \to \mathbb{R}^+$, α is *Min-sum preferred* (with respect to f) to β, written as $\alpha \leq_f \beta$, if and only if

— $\sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$ «

We also have the strict version of Min-sum preference, defined as follows.

Definition 2.35 » Strictly Min-sum preferred
For all $\alpha, \beta \in \mathcal{A}$, for some function $f : T \to \mathbb{R}^+$, α is *strictly Min-sum preferred* (with respect to f) to β, written as $\alpha <_f \beta$, if and only if

— $\sum_{i=1}^{m} f(\alpha_i) < \sum_{i=1}^{m} f(\beta_i)$ «

And we have the definition for Min-sum equivalent, as follows.

Definition 2.36 » Min-sum equivalent
For all $\alpha, \beta \in \mathcal{A}$, for some function $f : T \to \mathbb{R}^+$, α is *Min-sum equivalent* (with respect to f) to β, written as $\alpha \equiv_f \beta$, if and only if

— $\sum_{i=1}^{m} f(\alpha_i) = \sum_{i=1}^{m} f(\beta_i)$ «

A Min-sum optimal decision is defined as follows.

Definition 2.37 » Min-sum optimal
For some function $f : T \to \mathbb{R}^+$, decision $\alpha \in \mathcal{A}$ is *Min-sum optimal* (with respect to f) if and only if

— $\alpha \leq_f \beta$ for all $\beta \in \mathcal{A}$ «

For some $f : T \to \mathbb{R}^+$, let B_f denote the set of best decisions from $\mathcal{A}$ with respect to the Min-sum relation for that f, and let O_f denote the set of optimal decisions from $\mathcal{A}$ with respect to the Min-sum relation for that f. The Min-sum relation forms a total preorder on $\mathcal{A}$, so we have that $\mathrm{B}_f = \mathrm{O}_f \neq \emptyset$.

The Min-sum of weights relation extends the Sorted-Pareto dominance relation, so we have the following result.

Proposition 2.7 » Sorted-Pareto dominance and Min-Sum of weights

For a multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where $f : T \to \mathbb{R}^+$ and is monotonic with respect to scale T, then we have for all $\alpha, \beta \in \mathcal{A}$,

(i) $\alpha \preccurlyeq_{\mathrm{SP}} \beta \Rightarrow \alpha \leq_f \beta$

If f is strictly monotonic with respect to scale T, then we have:

(ii) $\alpha \prec_{\mathrm{SP}} \beta \Rightarrow \alpha <_f \beta$

(iii) $\mathrm{O}_{\mathrm{SP}} \supseteq \mathrm{O}_f$ ◇

Proof:

(i) Suppose that $\alpha \preccurlyeq_{\mathrm{SP}} \beta$, then for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$. Monotonicity of f implies that, for all $i \in \{1, \ldots, m\}$, $f(\alpha_{(i)}) \leq f(\beta_{(i)})$. This implies that

$\sum_{i=1}^{m} f(\alpha_{(i)}) \leq \sum_{i=1}^{m} f(\beta_{(i)})$, which is equivalent to $\sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$, and by definition of $\leq_f$ gives us that $\alpha \leq_f \beta$, proving the result.

(ii) Suppose that $\alpha \prec_{\text{SP}} \beta$, then for all $i \in \{1,\ldots,m\}$, $\alpha_{(i)} \leq \beta_{(i)}$, and exists $j \in \{1,\ldots,m\}$ such that $\alpha_{(i)} < \beta_{(i)}$. Strict monotonicity of f implies that $f(\alpha_{(j)}) < f(\beta_{(j)})$. This implies that $\sum_{i=1}^{m} f(\alpha_{(i)}) < \sum_{i=1}^{m} f(\beta_{(i)})$, which is equivalent to $\sum_{i=1}^{m} f(\alpha_i) < \sum_{i=1}^{m} f(\beta_i)$, and by definition of $<_f$ gives us that $\alpha <_f \beta$, proving the result.

(iii) This follows from (ii) and Proposition 2.1. ▪

We will discuss further the relationship between the Min-sum of weights relation and Sorted-Pareto dominance later in Section 5.3.

2.6.6 Sorted-Pareto and Ordered Weighted Averages

Ordered weighted averages (OWA) [Yag88] are a family of aggregation operator used for multicriteria decision making for maximising positive preferences. The scale of preference values used is the unit interval, i.e., we have that $T = [0,1]$, and these values indicate degrees of satisfaction for a criterion, which constitute the preference vector for each decision. The OWA operators also incorporate a *weighting vector* $\mathbf{w} = (W_1, W_2, \ldots, W_m)$, which assign weights to the values in the sorted preference vectors. Then for some OWA operator $F_{\mathbf{w}}$ with weighting vector $\mathbf{w}$, we have that $F_{\mathbf{w}}(\alpha) = W_1\alpha_{(1)} + W_2\alpha_{(2)} + \cdots + W_m\alpha_{(m)}$. The preference relation is given as follows:

Definition 2.38 » Ordered Weighted Averages
For all $\alpha, \beta \in \mathcal{A}$, for some OWA operator $F_{\mathbf{w}}$ with weighting vector $\mathbf{w}$, α is $F_{\mathbf{w}}$ *preferred* to β, written as $\alpha \succeq_{F_{\mathbf{w}}} \beta$, if and only if

— $F_{\mathbf{w}}(\alpha) \geq F_{\mathbf{w}}(\beta)$ «

We can also define the strict relation $\succ_{F_{\mathbf{w}}}$ and the equivalence relation $\equiv_{F_{\mathbf{w}}}$ in the obvious ways, where $\alpha \succ_{F_{\mathbf{w}}} \beta$ if and only if $F_{\mathbf{w}}(\alpha) < F_{\mathbf{w}}(\beta)$, and $\alpha \equiv_{F_{\mathbf{w}}} \beta$ if and only if $F_{\mathbf{w}}(\alpha) = F_{\mathbf{w}}(\beta)$. The OWA relation is a total preorder on $\mathcal{A}$, and the set of optimal decisions is denoted by $\text{O}_{F_{\mathbf{w}}}$.

As we can see the resulting relation $\succeq_{F_{\mathbf{w}}}$ depends on the weighting vector $\mathbf{w}$, for example, if we choose $\mathbf{w}$ such that $W_1 = 1$ and $W_i = 0$ for $i \in \{2,\ldots,m\}$, then that gives us the Maximin operator as previously defined.

Ordered Weighted Averages extend the Sorted-Pareto dominance relation, so we have the following result.

Proposition 2.8 » Sorted-Pareto and Ordered Weighted Averages

For a multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where T is a subset of the reals and $\mathbf{w}$ is such that for all $i \in \{1, \ldots, m\}$, $W_i \geq 0$, then we have for all $\alpha, \beta \in \mathcal{A}$,

(i) $\alpha \succcurlyeq_{\text{SP}} \beta \Rightarrow \alpha \succeq_{F_{\mathbf{w}}} \beta$

If, for all $i \in \{1, \ldots, m\}$, $W_i > 0$, then

(ii) $\alpha \succ_{\text{SP}} \beta \Rightarrow \alpha \succ_{F_{\mathbf{w}}} \beta$

(iii) $\mathrm{O}_{\text{SP}^*} \supseteq \mathrm{O}_{F_{\mathbf{w}}}$ ◇

Proof:

(i) Suppose $\alpha \succcurlyeq_{\text{SP}} \beta$. Then by definition we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \geq \beta_{(i)}$. Since for all $i \in \{1, \ldots, m\}$, $W_i \geq 0$, then we have for all $i \in \{1, \ldots, m\}$, $W_i \alpha_{(i)} \geq W_i \beta_{(i)}$, and therefore $\sum_{i=1}^{m} W_i \alpha_{(i)} \geq \sum_{i=1}^{m} W_i \beta_{(i)}$. By definition, this gives us $F_{\mathbf{w}}(\alpha) \geq F_{\mathbf{w}}(\beta)$, and therefore we have that $\alpha \succeq_{F_{\mathbf{w}}} \beta$.

(ii) Suppose $\alpha \succ_{\text{SP}} \beta$. Then by definition we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \geq \beta_{(i)}$ and there exists $j \in \{1, \ldots, m\}$ such that $\alpha_{(j)} > \beta_{(j)}$. Since for all $i \in \{1, \ldots, m\}$, $W_i > 0$, then we have for all $i \in \{1, \ldots, m\}$, $W_i \alpha_{(i)} \geq W_i \beta_{(i)}$, and there exists some $j \in \{1, \ldots, m\}$ such that $W_j \alpha_{(j)} > W_j \beta_{(j)}$. Therefore this gives us that $\sum_{i=1}^{m} W_i \alpha_{(i)} > \sum_{i=1}^{m} W_i \beta_{(i)}$ and by definition, we have $F_{\mathbf{w}}(\alpha) \geq F_{\mathbf{w}}(\beta)$, and therefore $\alpha \succ_{F_{\mathbf{w}}} \beta$.

(iii) This follows from (ii) and from Proposition 2.1. ∎

Ordered Weighted Averages extend the Sorted-Pareto dominance relation, however as seen with Generalized Lorenz dominance and the other relations in this section, Ordered Weighted Averages requires that the values in the preference vectors to be on a quantitative scale, since it performs multiplication of the weights in the weighting vector with the preference values.

2.6.7 Sorted-Pareto and Generalised Lorenz Dominance

The Generalised Lorenz Dominance relation [Atk70, Sho83] was developed in the context of social welfare distributions to compare the *equitability* of income distributions, i.e., the fairness of the values in positive preference vectors. The Lorenz curve of a distribution (or decision) is constructed as follows, considering the sorted preference vector associated with the decision.

Definition 2.39 » Lorenz curve of a decision
For a decision $\alpha \in \mathcal{A}$, with the mean of the distribution equal to μ, let $L(\alpha, p)$ be the *Lorenz curve* of α, where rational $p \in [0, 1]$, and

— $L(\alpha, \frac{k}{m}) = \sum_{i=1}^{k} \frac{\alpha_{(i)}}{m\mu}$, for $k = 1, \ldots, m$. «

Let $L(\alpha, 0) = 0$, and to create the curve, adjacent points, of the form $[\frac{k}{m}, L(\alpha, \frac{k}{m})]$, are joined together. For two distributions or decisions of equal means, i.e., where the sum of the preference vectors are equal, then we have the following.

Definition 2.40 » Lorenz dominance
For $\alpha, \beta \in \mathcal{A}$, α *Lorenz-dominates* β, written as $\alpha \succcurlyeq_{\text{LOR}} \beta$, if and only if

— $L(\alpha, p) \geq L(\beta, p)$, for all rational $p \in [0, 1]$ «

This means that if the Lorenz curve of α lies above the Lorenz curve of β, then we have that α is preferred to β as it is more equitable.

To compare decisions that do not have the same mean, then we have the notion of Generalized Lorenz dominance, where a Generalized Lorenz curve is given by scaling up the regular Lorenz curve by the mean of the distribution, i.e.,

Definition 2.41 » Generalized Lorenz curve of a decision
For a decision $\alpha \in \mathcal{A}$, let $GL(\alpha, p)$ be the generalized Lorenz curve of α, where rational $p \in [0, 1]$ and $GL(\alpha, p) = \mu L(\alpha, p)$, and

— $GL(\alpha, \frac{k}{m}) = \frac{1}{m} \sum_{i=1}^{k} \alpha_{(i)}$, for $k = 1, \ldots, m$. «

This gives us the Generalized Lorenz dominance relation as follows.

Definition 2.42 » Generalized Lorenz dominance
For $\alpha, \beta \in \mathcal{A}$, α *Generalized Lorenz dominates* β, written as $\alpha \succcurlyeq_{GL} \beta$, if and only if

— $GL(\alpha, \frac{k}{m}) \geq GL(\beta, \frac{k}{m})$, for $k = 1, \ldots, m$. «

We can also define the strict relation $\succ_{GL}$ in terms of $\succcurlyeq_{GL}$, where $\alpha \succ_{GL} \beta$ if and only if $\alpha \succcurlyeq_{GL} \beta$ and $\beta \not\succcurlyeq_{GL} \alpha$. We define the equivalence relation $\equiv_{GL}$ by, $\alpha \equiv_{GL} \beta$ if and only if $\alpha \succcurlyeq_{GL} \beta$ and $\beta \succcurlyeq_{GL} \alpha$. Generalized Lorenz dominance is a total preorder on $\mathcal{A}$, and the set of optimal elements of $\mathcal{A}$ with respect to Generalized Lorenz dominance, is denoted by O_{GL}.

Generalized Lorenz dominance extends Sorted-Pareto dominance, so we have the following result.

Proposition 2.9 » Sorted-Pareto and Generalized Lorenz dominance

For all $\alpha, \beta \in \mathcal{A}$, we have:

(i) $\alpha \succcurlyeq_{SP} \beta \Rightarrow \alpha \succcurlyeq_{GL} \beta$

(ii) $\alpha \succ_{SP} \beta \Rightarrow \alpha \succ_{GL} \beta$

(iii) $O_{SP^*} \supseteq O_{GL}$ ◇

Proof:

(i) Suppose $\alpha \succcurlyeq_{SP} \beta$. Then by definition we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \geq \beta_{(i)}$. Therefore, for $k = 1, \ldots, m$, we have $\frac{1}{m}\sum_{i=1}^{k} \alpha_{(i)} \geq \frac{1}{m}\sum_{i=1}^{k} \beta_{(i)}$, which gives us $GL(\alpha, \frac{k}{m}) \geq GL(\beta, \frac{k}{m})$, for $k = 1, \ldots, m$. Therefore we have that $\alpha \succcurlyeq_{GL} \beta$.

(ii) Suppose $\alpha \succ_{SP} \beta$. By (i), we have that $\alpha \succcurlyeq_{GL} \beta$. By definition of $\succ_{SP}$ we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \geq \beta_{(i)}$ and there exists $j \in \{1, \ldots, m\}$ such that $\alpha_{(j)} > \beta_{(j)}$. Therefore there exists $j \in \{1, \ldots, m\}$, such that $\frac{1}{m}\sum_{i=1}^{j-1} \alpha_{(i)} \geq \frac{1}{m}\sum_{i=1}^{j-1} \beta_{(i)}$, and $\frac{1}{m}\sum_{i=j}^{m} \alpha_{(i)} > \frac{1}{m}\sum_{i=j}^{m} \beta_{(i)}$. Therefore we have that $\beta \not\succcurlyeq_{GL} \alpha$.

(iii) This follows from (ii) and from Proposition 2.1. ∎

Generalized Lorenz Dominance extends Sorted-Pareto dominance, however it requires that the values in the preference vectors (i.e., the income distribution) to be

on a quantitative or additive scale, since it performs a summation of the preference values to construct the Generalized Lorenz curve.

2.6.8 Sorted-Pareto and Minimax Regret

We have seen how Sorted-Pareto dominance is related to other preference relations that use a reordering of the preference vectors. Now we examine a preference relation that maintains the ordering of the preference vectors, and we compare it with Sorted-Pareto dominance. The Minimax Regret [Sav51, BPPS06, LS82] relation looks to minimise the worst case *regret*, where the regret of a decision with respect to another decision is the difference between the maximum preference values, and the *maximum regret* of a decision is the maximum regret over all decisions. The Minimax Regret relation is used in situations where decision making is under uncertainty, so for a multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where T is here assumed to be a subset of the reals, the set of decision aspects $\mathcal{S}$ corresponds to the set of possible scenarios that can occur. Therefore the comparison between decisions occurs on the preference vectors where the ordering is maintained according to the scenarios. Given this we define the notion of regret.

Definition 2.43 » Regret of α with respect to β
For $\alpha, \beta \in \mathcal{A}$, the *regret* of α with respect to β, denoted by $R(\alpha, \beta)$ is

— $R(\alpha, \beta) = \max_{i \in \mathcal{S}} \beta_i - \alpha_i$ «

Now we define the notion of maximum regret of a decision.

Definition 2.44 » Maximum regret of a decision
For $\alpha \in \mathcal{A}$, the *maximum regret* of α, denoted by $MR(\alpha, \mathcal{A})$ is

— $MR(\alpha, \mathcal{A}) = \max_{\beta \in \mathcal{A}} R(\alpha, \beta)$ «

To compare two decision based on their maximum regret, we have the following.

Definition 2.45 » Maximum regret preferred
For $\alpha, \beta \in \mathcal{A}$, α is *maximum regret preferred* to β, if and only if

— $MR(\alpha, \mathcal{A}) \leq MR(\beta, \mathcal{A})$ «

Given this definition, the decisions that are optimal with respect to maximum regret, are those that have the minimal maximum regret, and we denote this set as O_{MR}. Since the comparison performed by Minimax regret involved maintaining the ordering of the states or aspects, i.e., the preference vector is not sorted, then the relation is not comparable with Sorted-Pareto dominance. Let us look at an example.

Example 2.6 ► Sorted-Pareto and Minimax regret**.**
Consider some multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where we have two decisions α and β which are evaluated over four different aspects, and the preference vectors are given as follows:

- $v(\alpha) = (3, 1, 4, 5)$
- $v(\beta) = (1, 4, 3, 4)$

We can see that the regret of α with respect to β is 3, which occurs in scenario 2, and the regret of β with respect to α is 2, which occurs in scenario 1. This means that $MR(\beta, \mathcal{A}) < MR(\alpha, \mathcal{A})$, and we have that

- $O_{MR} = \{\beta\}$

If we evaluate the decisions using Sorted-Pareto dominance, we have

- $v(\alpha)^{\uparrow} = (1, 3, 4, 5)$
- $v(\beta)^{\uparrow} = (1, 3, 4, 4)$

We can see that $\alpha \succ_{\text{SP}} \beta$ and we have that

- $O_{\text{SP}^*} = \{\alpha\}$

We can also see that O_{MR} is incomparable with O_{SP^*}. ▲

2.6.9 Other Relations using Additional Information

There are other preference relations that use information not considered by the Sorted-Pareto dominance relation. We may have additional information in relation to the decision aspects, for example, there may be a *probability distribution P* on the set of aspects $\mathcal{S}$, which give a weighting to each preference value in the preference vector for each decision. A probability distribution P over $\mathcal{S}$ is a mapping from $\mathcal{S} \to [0, 1]$ such that $P(i) \geq 0$ for all $i \in \mathcal{S}$, and $\sum_{i \in \mathcal{S}} P(i) = 1$. This could represent a situation where the aspects correspond to different possible states of the world, of

which one will actualise, and the probabilities over the aspects are the likelihood of each state occurring.

For example, if we consider positive preference values, e.g., utilities, then given a probability distribution P we can try and maximise our *expected* utility, such as in the Expected Utility preference relation [Sav54], where α is *EU-preferred* to β, if and only if $\sum_{i \in \mathcal{S}} P(i)\alpha_i \geq \sum_{i \in \mathcal{S}} P(i)\beta_i$, and $P(i)$ is the probability of state $i \in \mathcal{S}$ occurring. However in this thesis, since we consider situations where this probability information is not available, we do not discuss further these types of relations.

2.7 Chapter Conclusion

In this chapter, we provided some introductory material to give some background to the work in the thesis. We gave an introduction to preference and preference relations, firstly in a general context and also in the context of a decision problem where we have multiple preferences associated with each decision. We looked at the resulting preferred elements of a set of decisions and discussed the relationships between them. We also looked at the notion of an extension to a relation, and examined the relationships between the sets of optimal elements with respect to a given relation and its extension.

We introduced Sorted-Pareto dominance, which is a qualitative preference relation that extends Pareto-dominance, and is one of the major focuses of this thesis. We discussed the relationship between Sorted-Pareto dominance and other quantitative preference relations for minimising negative preferences and maximising positive preferences. We also looked at relations that are connected to Sorted-Pareto based on the comparison of reordered preference vectors, and relations that utilise additional quantitative information. We explore further the Sorted-Pareto dominance relation in Chapter 5.

Chapter 3

Background II - Soft Constraints

3.1 Introduction

A combinatorial problem involves a set of decision variables, where each variable can take values from their given domains, and solving a combinatorial problem involves finding one or more solutions consistent with the problem description, where a solution is an assignment of domain values to the decision variables. The constraint programming paradigm is an approach that naturally fits the solving of combinatorial problems, where constraints are used to model the problem by specifying the relationships between decision variables and the values that the variables are allowed to take. Hard constraints formalisms can be used to model problems where the problem description specifies properties that must be present in a solution to the problem, i.e., required properties, and these properties are modelled using *hard* constraints – where a hard constraint must be *satisfied* by a solution for the requirement modelled by the constraint to be met.

However there are a number of situations where modelling a problem using only hard constraints can be inadequate. These include:

- when a problem is *over-constrained*, i.e., that all hard constraints cannot be satisfied by any solution, and therefore there are no solutions to the problem that meet all requirements.
- when a problem is *under-constrained*, i.e., a large number of solutions satisfy the hard constraints and thus meet the problem requirements – this may be prohibitive in terms of generating the solutions to the problem, or once the set has been generated, the choice between decisions is inconsequential since all solutions are equivalent.
- when the problem may have constraints that can be partially satisfied, i.e., a given constraint can have different levels of satisfaction
- when the problem may have requirements that are optional or *preferential* (rather than absolute requirements), meaning that a solution may or may not meet these requirements and still be a valid solution to the problem.
- when there is uncertainty around the problem, e.g., there could be uncertainty as to whether or not a constraint is satisfied.
- when the user can specify their own preferences or desires in the problem over the set of consistent solutions to the problem.

In these situations above, we can use soft constraints to effectively model these

problems. Soft constraints are a generalisation of hard constraints, where instead of a constraint either being satisfied or not satisfied by an assignment to a set of variables, as in hard constraints, a soft constraint associates with an assignment a preference value. The values associated to each assignment can then be combined and reasoned with to deduce information about the assignment, to compare assignments with one another to see which is preferable, and to choose the most preferred solution or set of solutions to the problem.

In this chapter, we give a brief introduction to constraints formalisms with a view to providing some background for the works of the thesis. The chapter outline is as follows. In Section 3.2 we give some background on hard constraints and constraint satisfaction problems (CSPs). In Section 3.3 we then look at soft constraints, which allow us to specify preferences in a constraint problem, and we review some of the frameworks for soft constraints in the literature that are relevant to our work. In Section 3.4 we describe a general constraints problem, which gives the background to our work in later chapters and in Section 3.5 we look at some general algorithm such as backtracking search and branch and bound search which are used to solve problems of this nature.

3.2 Hard Constraints

In this section, we give a brief overview of hard constraints and hard constraint networks. Firstly we give some preliminary definitions and notation.

In a combinatorial problem, a *variable* represents an unknown value from a finite set of values, known as its *domain*. For some variable X, let $\mathcal{D}(X)$ denote the domain of variable X, and for some sequence of variables $V = \{X_1, \ldots, X_n\}$, let $\mathcal{D}(V) = \prod_{X_i \in V} \mathcal{D}(X_i)$ denote the cartesian product of the domains of the variables.

For a sequence of variables $V = \{X_1, \ldots, X_n\}$ an *assignment* is an ordered tuple of domain values of the variables in V, i.e., an assignment is a tuple in $\mathcal{D}(V)$.

For an assignment $t \in \mathcal{D}(V)$, and set of variables W, where $W \subseteq V$, the *projection* of an assignment $t \in \mathcal{D}(V)$ over the set W, denoted by $t^{\downarrow W}$, is the subtuple of t over the variables in W. We also call this subtuple $t^{\downarrow W}$ a *scoped tuple*, i.e., the tuple t is scoped with respect to the set of variables W.

Now we look at a *hard constraint network*, which is defined as follows.

Definition 3.1 » Hard constraint network (HCN)
A *hard constraint network* is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H \rangle$, where:

- $\mathcal{X}$ is a set of n variables, $\{X_1, \ldots, X_n\}$.
- $\mathcal{D}$ is a set of variable domains, $\{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$.
- $\mathcal{C}_H$ is a set of hard constraints, where a hard constraint $c_V \in \mathcal{C}_H$ with scope $V \subseteq \mathcal{X}$ is a relation on V, i.e., $c_V \subseteq \mathcal{D}(V)$. «

Given a hard constraint $c_V \in \mathcal{C}_H$ with scope V, and given a set of variables $W \subseteq \mathcal{X}$, with some assignment $t \in \mathcal{D}(W)$, we have the following terminology associated with hard constraint c_V:

- c_V is *completely assigned* by t if $V \subseteq W$.
- c_V is *satisfied* by t, (or t *satisfies* c_V), if $t^{\downarrow V} \in c_V$.

Given an assignment $t \in \mathcal{D}(W)$, where $W \subseteq \mathcal{X}$, we have the following terminology associated with assignment t:

- t is *consistent* if it satisfies all $c_V \in \mathcal{C}_H$ that are completely assigned by t.
- t is a *complete* assignment if $W = \mathcal{X}$; inversely t is a *partial* assignment if it is not complete, i.e., if $W \subset \mathcal{X}$.
- t is a *solution* to the hard constraint network if it is complete and consistent.

We define the set of all consistent solutions to a hard constraint network as follows:

Definition 3.2 » Solution set (HCN)
The *solution set* of a hard constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H \rangle$, denoted by $\text{Sol}(P)$, is given by

— $\text{Sol}(P) = \{t \in \mathcal{D}(\mathcal{X}) : t \text{ satisfies } c_V, \text{ for all } c_V \in \mathcal{C}_H\}$. «

That is, $\text{Sol}(P)$ is the set of all complete assignments that are consistent with the hard constraints of the problem.

Finally, A *Constraint Satisfaction Problem* (CSP) is a task on a hard constraint network, which could be, to find one solution, or to find all solutions of the network.

3.3 Soft Constraints

For situations where hard constraints are not expressive enough, such as those given in Section 3.1, we can use soft constraints to model these types of problems. A hard constraint is a relation which specifies which scoped tuples of domain values satisfy the constraint, whereas a *soft constraint* is a function which associates a *preference value* or preference degree to scoped tuples of domain values. The preference values associated with a solution can then be combined to give an overall preference level of that solution, and then solutions can be compared to see which are preferred to others, and which solutions are optimal. In this section we look briefly at some soft constraints formalisms, with a view to providing some background for the works in the thesis.

3.3.1 Semiring Constraint Network

One of the most well known formalisms for soft constraints is the *semiring constraint network* [BMR97] which is based on an algebraic structure called a c-semiring which can be used to model and reason with soft constraints. The semiring constraint network is defined as follows.

Definition 3.3 » Semiring constraint network (SCN)
A *semiring constraint network* [BMR97] is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, where

- $\mathcal{X}$ is a set of n variables, $\{X_1, \ldots, X_n\}$.
- $\mathcal{D}$ is a set of variable domains, $\{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$.
- S is a c-semiring, which is a tuple $\langle E, +, \times, 0, 1 \rangle$ where
  - E is a set of values, where $0 \in E$ and $1 \in E$.
  - $+$ is an operator closed in E, which is associative, commutative, idempotent, and for which 0 is a neutral element and 1 is an annihilator.
  - $\times$ is an operator closed in E, which is associative, commutative, and for which 0 is an annihilator and 1 a neutral element.
  - $\times$ distributes over $+$.
- $\mathcal{C}_S$ is a finite set of soft constraints, where each soft constraint $s_V \in \mathcal{C}_S$, with scope $V \subseteq \mathcal{X}$, associates c-semiring values from E to scoped tuples of domain values, i.e., $s_V : \mathcal{D}(V) \rightarrow E$. «

As in a hard constraint network, in a semiring constraint network we have a set of variables $\mathcal{X}$, where each variable X can take a value from their domain $\mathcal{D}(X)$. However, instead of a set of hard constraints we have a set of soft constraints $\mathcal{C}_S$ on the variables in the network, which associate semiring values to tuples of domain values. Given a soft constraint $s_V \in \mathcal{C}_S$ with scope $V \subseteq \mathcal{X}$, and given a set of variables $W \subseteq \mathcal{X}$, with some assignment $t \in \mathcal{D}(W)$, and when s_V is completely assigned by t, i.e., $V \subseteq W$, then the preference level, i.e., a value from E, associated to t by s_V is denoted by $s_V(t^{\downarrow V})$. We can also write this compactly as $s_V(t)$.

The $\times$ operator is used to combine semiring values, and the $+$ operator induces an ordering $\succeq_S$ over the values of the semiring, defined by, for all $a, b \in E$, $a \succeq_S b \iff a + b = a$. A c-semiring is *idempotent* if $\times$ is idempotent, i.e., for all $a \in E$, $a \times a = a$. A c-semiring is monotonic if $\times$ is monotonic, i.e., for all $a, b \in E$, $a \succeq_S b \Rightarrow a \times c \succeq_S b \times c$, and it is strictly monotonic if $\times$ is strictly monotonic, i.e., $a, b \in E$, $a \succeq_S b \iff a \times c \succeq_S b \times c$.

We now give the definition of the preference level of an assignment for a semiring constraint network.

Definition 3.4 » Assignment preference level (SCN)
For a semiring constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, the *preference level* $\rho(t)$ of an assignment $t \in \mathcal{D}(\mathcal{X})$ is given by

— $\rho(t) = \bigtimes_{s_V \in \mathcal{C}_S} s_V(t)$ «

That is, the preference level of t is the combination of all semiring values associated to t by the soft constraints. The preference relation for a semiring constraint network is defined as follows.

Definition 3.5 » Preference relation (SCN)
For a semiring constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, a solution $t \in \mathcal{D}(\mathcal{X})$ is *preferred* to solution $t' \in \mathcal{D}(\mathcal{X})$, if and only if

— $\rho(t) \preceq_S \rho(t')$ «

Given this preference relation, we can define an optimal solution to a semiring constraint network as follows.

Definition 3.6 » Optimal solution (SCN)
For a semiring constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, a solution $t \in \mathcal{D}(\mathcal{X})$ is *optimal* if and only if

— there is no $t' \in \mathcal{D}(\mathcal{X})$ such that $\rho(t)' \succeq_S \rho(t)$ and $\rho(t) \not\succeq_S \rho(t')$ «

That is, there is no other complete assignment t' such that t' is strictly preferred to t. We can see that this definition for an optimal solution of a semiring constraint network is consistent with the general definition for an optimal element of a preference relation (Definition 2.9).

Another well known general formalism which is related to the semiring constraint network is the *Valued Constraint Network*, [SFV95], where the semiring structure in the network is replaced by a *valuation structure*, based on totally ordered degrees of preference; the valued constraint network can be used to represent the same types of networks as the semiring constraint network [BMR+99], and has similar concepts for combining and comparing values. Both general formalisms can be used to model a number of different type of soft constraints formalisms, such as weighted constraints, which we look at next.

3.3.2 Weighted Constraints

In weighted constraints, the values or *weights* associated to scoped tuples by a weighted constraint represent the *cost* of the tuple. In k-weighted constraints [SH81], the maximum weight that can be specified is denoted by k, and any tuple with weight k does not satisfy the constraint. We give the definition of a *k-weighted constraint network* as follows.

Definition 3.7 » k-weighted constraint network (k-WCN)
A *k-weighted constraint network* is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}_W, k \rangle$, where

- $\mathcal{X}$ is a set of n variables, $\{X_1, \ldots, X_n\}$,
- $\mathcal{D}$ is a set of variable domains, $\{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$,
- $\mathcal{C}_W$ is a set of weighted constraints, where each weighted constraint $w_V \in \mathcal{C}_W$ with scope $V \subseteq \mathcal{X}$, associates a value from $[0, k]$ to scoped tuples of domain values, i.e., $w_V : \mathcal{D}(V) \rightarrow [0, k]$.
- k is an element in $\mathbb{N} \cup \{\infty\}$. «

We now define the preference level, or *cost*, of an assignment in a k-weighted constraint network.

Definition 3.8 » Assignment preference level (k-WCN)
For a k-weighted constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_W, k \rangle$, the *preference level* $\rho(t)$ of an assignment $t \in \mathcal{D}(\mathcal{X})$ is given by

— $\rho(t) = \sum_{w_V \in \mathcal{C}_W} w_V(t)$ «

That is, the cost of an assignment t is the sum of the weights associated to t by the weighted constraints. Given that k denotes a cost that is prohibited in a weighted constraint network, then an assignment t that has a cost equal to or higher than k does not satisfy the problem, i.e., we have that for $t \in \mathcal{D}(\mathcal{X})$, if $\rho(t) \geq k$, then $t \notin \text{Sol}(P)$.

The associated preference relation to a k-weighted constraint network is defined as follows.

Definition 3.9 » Preference relation (k-WCN)
For a k-weighted constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_W, k \rangle$, a solution $t \in \text{Sol}(P)$ is *preferred* to solution $t' \in \text{Sol}(P)$, if and only if

— $\rho(t) \leq \rho(t')$ «

An optimal solution to a k-weighted constraint network is defined as follows.

Definition 3.10 » Optimal solution (k-WCN)
For a k-weighted constraint network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_W, k \rangle$, a solution $t \in \text{Sol}(P)$ is *optimal*, if and only if

— for all $t' \in \text{Sol}(P)$, $\rho(t) \leq \rho(t')$ «

We can see that this definition for an optimal solution of a k-weighted constraint network is consistent with the general definition for an best element of a preference relation (Definition 2.8), however since the scale of preference levels $[0, k]$ is totally ordered by $\leq$, then the k-weighted preference relation is complete, and we have from Remark 2.3 (iii) that the best elements and the optimal elements coincide.

3.4 A General Constraints Problem

For the purposes of the work in later chapters, we now give a definition of a constraints problem with hard and soft constraints. Firstly, we define a structure for handling the ordering and combination of preference degrees, which is called a Preference Degree Structure [FRW10], and this structure determines the preference relation over the set of solutions in a given problem.

Definition 3.11 » Preference degree structure (PDS)
A *preference degree structure* (PDS) is a tuple $\langle I, \otimes, \preceq \rangle$, where

- I is a set of preference degrees;
- $\otimes$ is a commutative and associative operator, monotonic with respect to $\preceq$ (i.e., for $a, b, c \in I$, $a \preceq b \Rightarrow a \otimes c \preceq b \otimes c$), which is used to combine the preference degrees;
- $\preceq$ is a preorder relation on the set of degrees I. «

We assume a minimising context, so for relation $\preceq$, we have that for all $u, v \in I$, if $u \preceq v$, then u is preferred to v. For relation $\preceq$, we also have the associated strict preference relation $\prec$, where for all $u, v \in I$, $u \prec v$ if and only if $u \preceq v$ and $v \not\preceq u$, and we say that u is strictly preferred to v.

Now we give the following definition of a general constraints problem which we use as the constraints formalism for our work in this thesis.

Definition 3.12 » General constraints problem (GCP)
A *general constraints problem* (GCP) is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where

- $\mathcal{X}$ is a set of n variables, $\{X_1, \ldots, X_n\}$;
- $\mathcal{D}$ is a set of variable domains, $\{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$;
- $\mathcal{C}_H$ is a set of hard constraints;
- $\mathcal{C}_S$ is a multiset of soft constraints, where for all $s_V \in \mathcal{C}_S$, with scope $V \subseteq \mathcal{X}$, $s_V : \mathcal{D}(V) \rightarrow I$;
- $\mathcal{P}$ is a preference degree structure $\langle I, \otimes, \preceq \rangle$. «

We now define the preference level of an assignment in a general constraints problem.

Definition 3.13 » Assignment preference level (GCP)
For a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where $\mathcal{P} = \langle I, \otimes, \preceq \rangle$, the *preference level* $\rho(t)$ of a solution $t \in \text{Sol}(P)$ is given by

— $\rho(t) = \bigotimes_{s_V \in \mathcal{C}_S} s_V(t)$ «

That is, the preference level of t is the combination of all the preference degrees associated to t by the soft constraints. The preference relation for a general constraints problem is defined as follows.

Definition 3.14 » Preference relation (GCP)
For a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where $\mathcal{P} = \langle I, \otimes, \preceq \rangle$, a solution $t \in \text{Sol}(P)$ is *preferred* to solution $t' \in \text{Sol}(P)$, if and only if

— $\rho(t) \preceq \rho(t')$ «

An optimal solution of a general constraints problem is defined as follows.

Definition 3.15 » Optimal solution (GCP)
For a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where $\mathcal{P} = \langle I, \otimes, \preceq \rangle$, a solution $t \in \text{Sol}(P)$ is *optimal* if and only if,

— there exists no $t' \in \text{Sol}(P)$ such that $\rho(t') \prec \rho(t)$. «

As discussed in Section 3.1, soft constraints are a generalisation of hard constraints and can also be used to express hard constraints, for example, as in k-weighted constraints, where for a constraint w_V, a tuple t with cost k does not satisfy the constraint, and therefore constraint w_V is a hard constraint. However in our General Constraints Problem we have an explicit separation of the hard constraints and soft constraints of the problem.

The general constraints problem as defined is similar to, but more general than the semiring constraint network as given in Section 3.3. In the preference degree structure, we do not assume a top or bottom element, and we do not assume the semiring addition operator $+$. Any c-semiring can be mapped to a preference degree structure, and so any semiring constraint (and any semiring constraint network) can be mapped to a general constraint/general constraint problem. So for some c-semiring $\langle E, +, \times, 0, 1 \rangle$, we have a preference degree structure $\langle I, \otimes, \preceq \rangle$ where:

- $\mathcal{I} = E$

- $\otimes = \times$
- $\preceq$ is given by: $\forall a, b \in E, b \succeq_s a \Leftrightarrow a \preceq b$

Therefore the ordering on solutions, and hence the set of optimal solutions, for both the c-semiring and the preference degree structure as defined is the same.

3.5 Searching for Solutions to Constraints Problems

In this section, we discuss some methods for searching for solutions to a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$. The space explored by the search is called a *search tree*, where a (non-leaf) *node* represents a variable $X \in \mathcal{X}$, and a downward *edge* from that node represents the assigning of a domain value $v \in \mathcal{D}(X)$ to variable X, i.e., the variable assignment (X, v). A *path* from the root to a given node represents an assignment to the set of variables along the path. A path from the root node to a leaf node represents a complete assignment or a solution to the problem, i.e., it is a tuple in $\mathcal{D}(\mathcal{X})$, and a path from the root to a non-leaf node represents a partial assignment to some strict subset of the problem variables.

For the purpose of the presentation in this section, we assume a fixed variable ordering of $(X_1, X_2, \ldots, X_n)$, i.e., that the variables are assigned in the given order during the search. Let us look at an example.

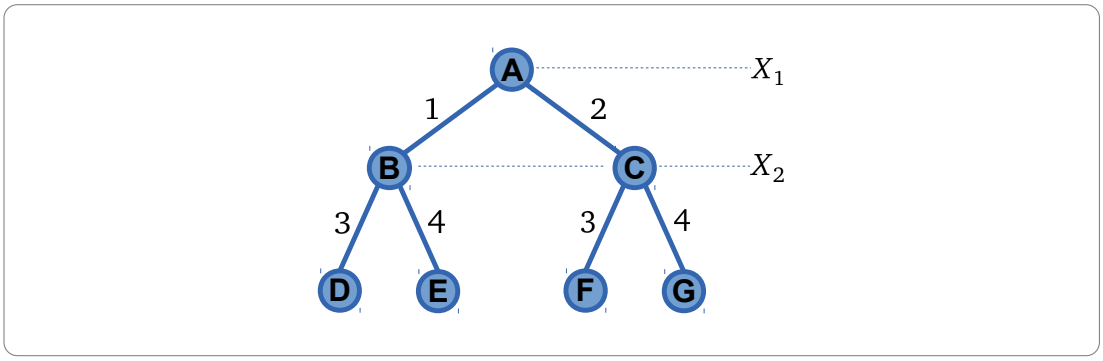


Figure 3.1: For Example 3.1, search tree for a problem with two variables X_1 and X_2, where $\mathcal{D}(X_1) = \{1, 2\}$ and $\mathcal{D}(X_2) = \{3, 4\}$.

Example 3.1 ► Search tree example**.**
Figure 3.1 shows an example search tree for two variables X_1 and X_2, where $\mathcal{D}(X_1) = \{1, 2\}$ and $\mathcal{D}(X_2) = \{3, 4\}$.

For example, node A *represents variable X_1, and the edge from node* A *to node* B *represents the variable assignment $(X_1, 1)$.*

We can also say that a given node represents the (partial or complete) assignment corresponding to the path from the root node to the given node.

For example, the assignment represented by node F, i.e., the path from root node A to F, is a complete assignment, denoted by $\{(X_1, 2), (X_2, 3)\}$*, and assuming the given variable ordering, this can be denoted by* $(2, 3)$*, which is a tuple in* $\mathcal{D}(\mathcal{X})$*.* ▲

3.5.1 Finding Consistent Solutions

First we consider searching for complete assignments that are consistent with the set of hard constraints in a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, i.e., the solutions in $\text{Sol}(P)$. One method to find all solutions to a constraints problem is to systematically *generate and test* every possible complete assignment, i.e., every tuple in $\mathcal{D}(\mathcal{X})$ is generated in turn and checked to see if it is consistent with the set of hard constraints. However this is a naive approach as it does not check that an assignment is inconsistent until the complete assignment is generated; it could be possible to detect inconsistencies at an earlier stage.

Backtracking Search

Backtracking search [DF98, Kum92] is a search procedure where each problem variable in turn is chosen and assigned a value from its domain, i.e., it is a *depth first search* of the search tree. During the search process, at any given node, if there are any constraints that are now completely assigned by the partial assignment represented at the given node, then the partial assignment is checked to see if it is valid with these constraints. If the search encounters such a node where the partial assignment is not valid, then the search will backtrack up the search tree and resume at the next node.

We look at a simple example for backtracking search. For ease of notation in the examples, we let c_{ij} be compact notation for a constraint c with scope $\{X_i, X_j\}$, and we let $\bar{c}_{ij}$ be the set of tuples such that, for all tuples $t \in \mathcal{D}(\{X_i, X_j\})$, if $t \notin c_{ij}$, then $t \in \bar{c}_{ij}$, i.e., $\bar{c}_{ij}$ is the set of tuples that are *not* allowed by the constraint c_{ij}.

Example 3.2 ► Backtracking search example**.**
Figure 3.2 shows a search tree for a problem with three variables X_1*,* X_2 *and* X_3*, where* $\mathcal{D}(X_1) = \{1, 2\}$*,* $\mathcal{D}(X_2) = \{3, 4\}$ *and* $\mathcal{D}(X_3) = \{5, 6\}$*.*

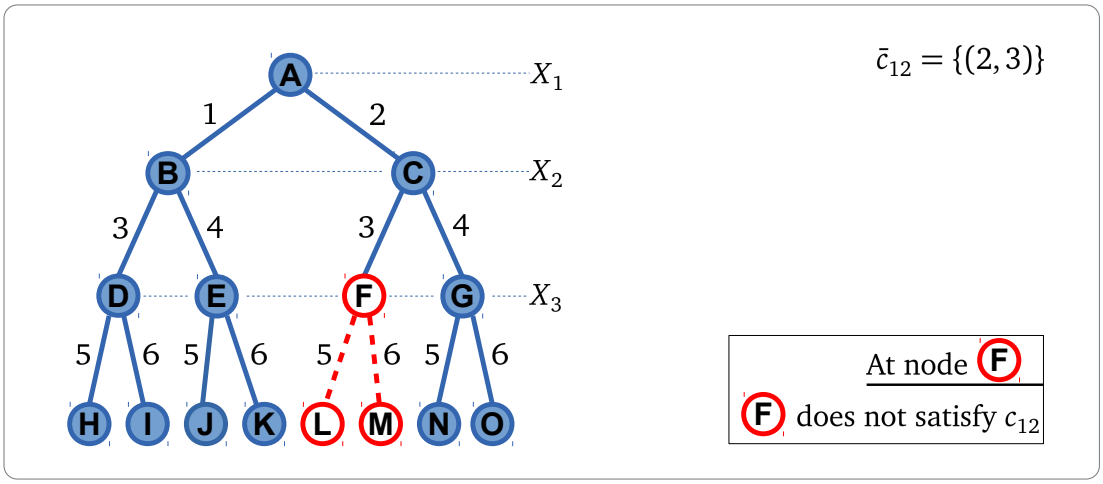


Figure 3.2: For Example 3.2, search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{1,2\}$, $\mathcal{D}(X_2) = \{3,4\}$ and $\mathcal{D}(X_3) = \{5,6\}$ and hard constraints c_{12} such that $\bar{c}_{12} = \{(2,3)\}$. A backtracking search will backtrack at the node labelled F.

Suppose we have one hard constraint c_{12}, such that $\bar{c}_{12} = \{(2,3)\}$, i.e., tuple $(2,3)$ is not allowed by the constraint c_{12}

For backtracking search, the search will backtrack at node (F), i.e., it will not visit nodes (L) and (M), since the partial assignment represented by node (F), i.e, tuple $(2,3)$, is not allowed by the hard constraint c_{12}. ▲

Local Consistency

As well as checking to see if the current partial assignment at some node is consistent with the hard constraints in the problem, the search procedure may also perform some techniques to remove inconsistent values from the variable domains. A *node consistency* algorithm checks to see if a variable domain is consistent with any *unary* constraints on that variable, and removes any domain values that are inconsistent. An *arc consistency* algorithm performs a similar task for pairs of variables in *binary* constraints. For example, in the *AC*3 algorithm [Mac77], for a binary constraint c_{ij} on variables (X_i, X_j), each domain value in $\mathcal{D}(X_i)$ is checked to see if there is a supporting value in $\mathcal{D}(X_j)$ such that the binary constraint is satisfied, and any inconsistent values in the domain of X_i are removed. Also, any revision of a variable domain will result in any other constraint with that variable in its scope to be checked again, since removing a value from a domain may trigger another value as inconsistent in another constraint. The algorithm continues in this manner until the whole constraint network is consistent. This technique for checking arc consistency can then be combined with backtracking search to remove inconsistent values from

domains of variables that have not yet been assigned, which is called *full look-ahead* [Dec03]. Let us look at an example.

Example 3.3 ► Look-ahead search example.
Figure 3.3 shows a search tree for a problem with three variables X_1, X_2 and X_3, and where the original domains of the variables are $\mathcal{D}(X_1) = \{1,2\}$, $\mathcal{D}(X_2) = \{3,4\}$ and $\mathcal{D}(X_3) = \{5,6\}$.

We also have two hard constraints c_{13} and c_{23}, such that $\bar{c}_{13} = \{(2,5)\}$ and $\bar{c}_{23} = \{(3,6)\}$, which specify the tuples that are not allowed by the constraints.

At node (C), the algorithm maintains arc consistency as follows:

(a) At node (C), the search algorithm will remove the value 5 from the domain of variable X_3, since it is inconsistent with the assignment $(X_1, 2)$.

(b) Since the domain of X_3 has been updated, the algorithm will check any other constraints that has X_3 in its scope, and as a result it will remove value 6 from the domain of X_3 since there is no supporting value in the domain of X_2 that satisfies constraint c_{23}.

(c) The algorithm continues and value 3 is removed from the domain of X_2, since there is no supporting value in the domain of X_3 that satisfies constraint c_{23}. ▲

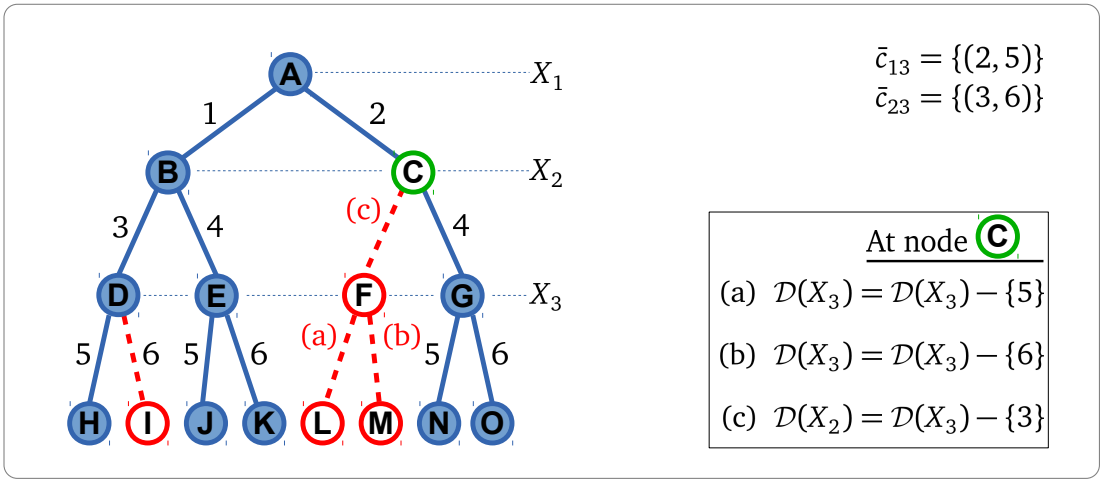


Figure 3.3: For Example 3.3, search tree for three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{1,2\}$, $\mathcal{D}(X_2) = \{3,4\}$ and $\mathcal{D}(X_3) = \{5,6\}$ and hard constraints $c_{13} = \{(2,5)\}$ and $c_{23} = \{(3,6)\}$. Maintaining arc consistency at the node labelled C will allow the search to prune part of the search below this node since there are no supported values.

3.5.2 Finding Optimal Solutions

In the previous section, we looked at searching for consistent solutions to a constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, i.e., the complete assignments that are consistent with $\mathcal{C}_H$. Now we look at searching for solutions that are more preferable, considering the set of soft constraints $\mathcal{C}_S$.

As previously defined in Section 3.3, a soft constraint $s_V \in \mathcal{C}_S$, with scope $V \subseteq \mathcal{X}$ associates values from a set I of preference degrees to scoped tuples of domain values, i.e., $s_V : \mathcal{D}(V) \rightarrow I$. The preference degrees given by the soft constraints are combined using the combination operator $\otimes$ to give the preference degree of a solution, i.e., for some $t \in \text{Sol}(P)$, $\rho(t) = \otimes_{s_V \in \mathcal{C}_S} s_V(t)$. Then solutions can be compared using the $\preceq$ operator, i.e., t is preferred to t' if $\rho(t) \preceq \rho(t')$, and an optimal solution is one such that there is no other solution t' such that $\rho(t') \prec \rho(t)$. We are looking to generate a set of optimal solutions to the problem P, denoted by O_P. Let us look at an example.

Example 3.4 ► Searching for optimal solutions example.
Figure 3.4 shows a search tree for a problem with three variables X_1, X_2 and X_3, and where the domains of the variables are $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$.

We have two soft constraints, s_{23} and s_{13}, which are defined in the following tables.

s_{12}		X_1: a	X_1: b
X_2	c	4	6
	d	3	2

s_{13}		X_1: a	X_1: b
X_3	e	4	1
	f	2	3

Each soft constraint associates different costs to scoped tuples.

For example, for tuple (a, c), the soft constraint s_{12} will associate a cost of 4 to this tuple, i.e., $s_{12}((a, c)) = 4$.

Suppose we wish to minimise the sum of these costs. Therefore the solutions with the smallest sum of costs are the ones that are optimal.

In Figure 3.4, the preference level of each solution is denoted by $\rho(t)$, which is the sum of all the costs associated to each solution by the constraints s_{12} and s_{13}.

Therefore we can see that the only optimal solution in O_P is the solution represented by node (N). ▲

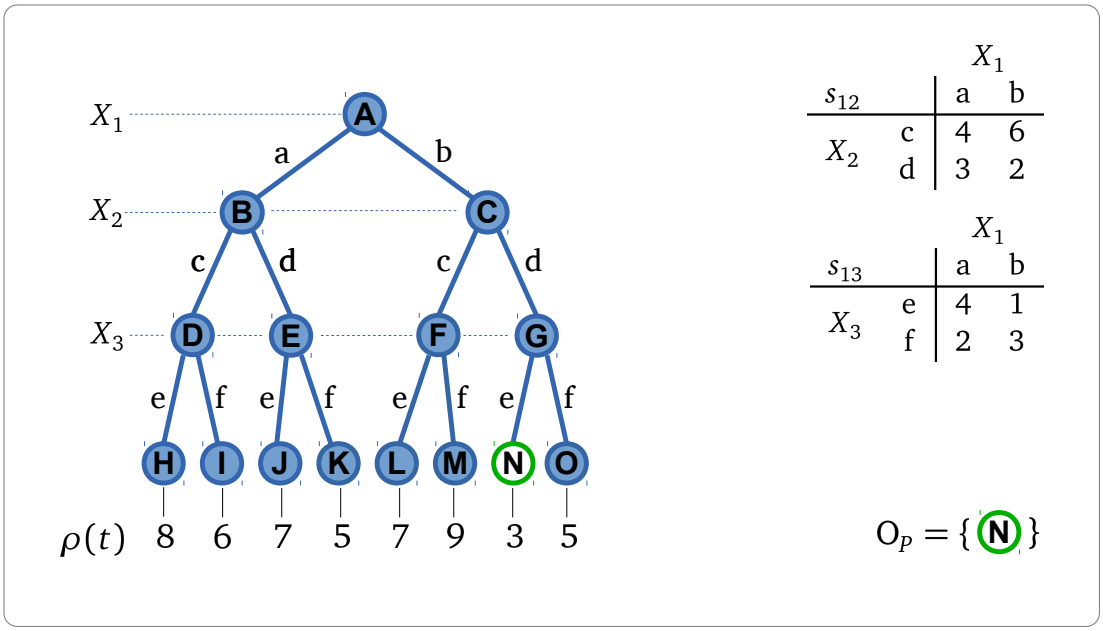


Figure 3.4: For Example 3.4, search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$, and the soft constraints s_{12} and s_{23} are given by the tables in the figure. The optimal solution is the one represented by the node labelled N.

Generate and Test

The generate and test approach described in Section 3.5.1 can also be used to generate the set of optimal solutions O_P for a problem P. Each tuple $t \in \mathcal{D}(\mathcal{X})$ is generated in turn and the preference level of the tuple t is checked against the preference level of every other solution $t' \in O_P$. If t is not dominated, then it is added to O_P. Since t might be strictly preferred to some other t' already in O_P, then any such t' is no longer optimal and is removed. As with the generate and test approach for hard constraints, this is a naive approach, as it could be possible to check if a solution is dominated without generating the complete assignment.

Branch and Bound

In a *branch and bound* search [Dec03, SFV95] for a constraints problem, the algorithm can also check if partial assignments are dominated by some solution already found. Again in this part we assume we are dealing with negative preference levels, e.g. costs. At each node in the search, a *lower bound* preference level $\rho_*(t)$ is generated for the partial assignment t represented by that node, which is a lower bound preference level of any complete assignment in the subtree under that node. If this lower bound is dominated by the preference level of any previously found

solution in O_P, then there is no complete assignment extending from that node that could be optimal, and the search can backtrack. This eliminates parts of the search tree that do not contain any optimal solutions.

There are a number of different methods for generating a lower bound at a particular node (for a survey, see [RvBW06, Ch. 9]). The simplest lower bound at a given node is given by just considering the preference levels of the soft constraints that are fully assigned at that node. For some partial assignment t let $T_s(t)$ be the set of soft constraints that are fully assigned by t. Then this simple lower bound preference level of t, denoted by $\rho^1_*(t)$, is given by

— $\rho^1_*(t) = \bigotimes_{s \in T_s(t)} s(t)$

That is, the lower bound is the combination of all the preference levels given by the soft constraints that are fully assigned by t.

As well as considering the soft constraints that are fully assigned at a given node, the lower bound can also take into account the soft constraints that are only partially assigned. For some partial assignment t, and some variable $X \in \mathcal{X}$ not yet assigned, let $t_{(X,v)}$ denote the extension of tuple t to include the assignment of domain value $v \in \mathcal{D}(X)$ to variable X, i.e, $t_{(X,v)} = t \cup (X, v)$. For some partial assignment t, let $P_s(t)$ be the set of soft constraints whose variables are partially assigned by tuple t. Then in the case of totally ordered preferences, another lower bound preference level of t, denoted by $\rho^2_*(t)$, is given by

— $\rho^2_*(t) = \rho^1_*(t) \otimes \min_{v \in \mathcal{D}(X)} \bigotimes_{s \in P_s(t) \cap T_s(t_{(X,v)})} s(t_{(X,v)})$

That is, the lower bound $\rho^2_*(t)$ is given by $\rho^1_*(t)$ combined with the minimum combination of preference levels given by the soft constraints that would be fully assigned by the extension of tuple t to include an assignment to variable X. In the case of partially ordered preferences, if the partial order is a *lattice*, where there is a greatest lower bound, then we can replace the min operation with the inf operation, which returns the greatest lower bound of the combination of preference levels. This approach used for ρ^2_* is similar to that used in the partial forward checking algorithm for partial constraint satisfaction [FW92].

Let us look at an example.

Example 3.5 ► Branch and Bound Search**.**
Figure 3.5 shows a search tree for the same problem as given in Example 3.4, where we have three variables X_1, X_2 and X_3, and where the domains of the variables are

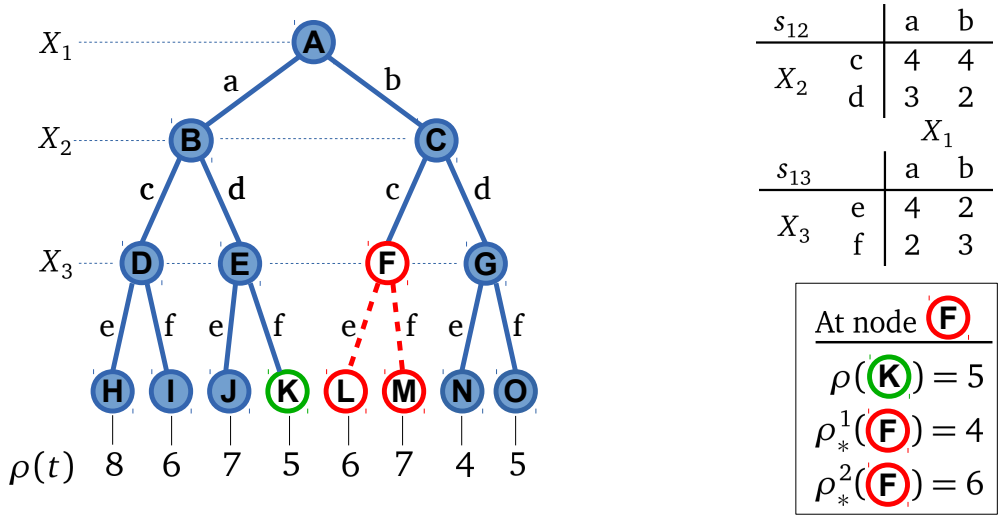


Figure 3.5: For Example 3.5, search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$, and the soft constraints s_{12} and s_{23} are given by the tables in the figure. The lower bound at the node labelled F is dominated by the preference level of a previously found solution.

$\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ *and* $\mathcal{D}(X_3) = \{e, f\}$.

We have two soft constraints, s_{23} *and* s_{13}*, which are defined in the following tables.*

s_{12}		X_1: a	b
X_2	c	4	4
	d	3	2

s_{13}		X_1: a	b
X_3	e	4	2
	f	2	3

We can see at node F *that we have a partial assignment* $t = (b, c)$*. The only optimal solution found at this point in the search is the solution represented by* K*, which is tuple* (a, d, f)*, and we have* $\rho(a, d, f) = 5$.

If we use lower bound ρ^1_**, then we have* $\rho^1_*(b, c) = 4$*, which is a lower bound preference level of any complete assignment in the subtree under node* F.

We can see that $\rho^1_*(b, c)$ *is not strictly dominated by* $\rho(a, d, f)$*, so in this case the search would continue to node* L.

If we use a better lower bound, i.e., ρ^2_**, then we have* $\rho^2_*(b, c) = 6$*, and in this case* $\rho^2_*(b, c)$ *is strictly dominated by* $\rho(a, d, f)$.

Hence the search can backtrack at node F *since it will not find any optimal solutions in the subtree under* F. ▲

The lower bound approach can be expanded further to factor in the preference levels of more constraints, such as the constraints that are completely unassigned at a given node, which is similar to that used in [Wal94, LMS99]. We will see such an approach later in Section 5.5.2. There are other types of lower bounds approaches, such as in the Russian Doll search [VLS96], which treats each node in the search as a subproblem, and uses the preference levels of optimal solutions for the subproblem in the subtree at a given node as a lower bound.

As well as depth first and branch and bound searches, there are other methods for solving such constraints problems, which include methods based on *complete inference* such as Bucket elimination [Dec99], which is a scheme that removes variables from the problem and adds implied constraints in order to make the problem easier to solve. However, since these complete inference schemes can be computationally expensive, Minibucket elimination [DR03] can be used as an approximation scheme and it is less computationally expensive. Also, the approximation generated can be used as a lower bound in an algorithm that combines both branch and bound search with inference. Other methods for solving these problems based on incomplete inference include soft local consistency [CS04, Sch00] which extends the classical notion of arc consistency as discussed in Section 3.5.1 to the soft constraint case, and is also used to generate a lower bound for a branch and bound algorithm.

3.6 Chapter Conclusion

In this chapter, we gave an introduction to hard and soft constraints problems. We describe the constraints formalism and associated notation used in the thesis, and look at some related formalisms for hard and soft constraints problems, such as semiring constraints problems and weighted constraints problems. We detailed some algorithms used to solve constraints problems of these types, such as depth first search and branch and bound algorithms, and we looked briefly at how to generate a lower bound for a branch and bound search. We will revisit some of these search algorithms in Chapter 5 as part of the works of the thesis.

Chapter 4

Qualitative Notions of Optimality

4.1 Introduction

In this chapter, we present a decision-making framework for qualitative decision making, which we call the Multiple-Ordering Decision Structure (MODS) framework, where we examine the various natural notions of optimality that occur and the relationships between these notions of optimality.

The setting we consider is where the task is to support a decision maker by presenting a subset of decisions from a larger set of initial decisions, where the decisions presented are *preferred* to the other decisions, or regarded as *better* in some way, thus aiding the decision maker by narrowing down the set of decisions from which they have to choose. We consider situations where there is more than one ordering on the decisions; this can occur, for example, in decision making under uncertainty, where there is more than one possible state that can happen and decisions are ordered differently depending on the state, or it can occur in multi-criteria decision making where evaluations in different criteria rank the decisions differently over the criteria.

We presuppose only qualitative preference information in relation to the set of decisions, i.e., that we only have a set of ordinal rankings on the decisions rather than any quantitative information, e.g., costs or utilities. However, the results given regarding the different notions of optimality also hold when there is this quantitative information available. Also, we do not assume any information on the relationships between different orderings, i.e., weighting or probabilistic information, such as, for example, criteria that are more important than others in a multi-criteria decision making problem, or states that are more likely to occur in a decision-making under uncertainty problem.

The chapter outline is as follows. In Section 4.2 we define the MODS framework and the different notions of optimality that we consider. Section 4.3 gives the main result of the chapter, which precisely describes the subclass relationships between the different notions of optimality. In Section 4.4, we show how these relationships simplify under three separate assumptions: (i) when there exists a necessarily optimal element; (ii) when there exists a best scenario for each decision; and (iii) when each scenario totally orders the set of decisions. In Section 4.5, we give a brief example of the MODS framework in multi-criteria decision making (MDCM). In Section 4.6, we look at how this framework and the subclass relationships between the optimality classes can be used to order the decisions for presenting to a decision maker, and we conclude the chapter with some discussion.

The framework developed in this chapter will also be used in the work in Chapter 6, where we develop an instance of the framework for the Sorted-Pareto dominance preference relation, which we introduced in Section 2.5.

4.2 Qualitative Notions of Optimality

In this section, we look at the Multiple-Ordering Decision Structure (MODS) framework, as originally given in [WO11]. We assume a minimising context, i.e., where smaller preference values are preferred, however an alternative framework for maximising preferences can easily be derived, for example, as originally presented in [WO11]. In the MODS framework, we have a set of elements, where each element in the set is interpreted as a decision (option, alternative, choice, etc.) that is available to a decision maker, and given this set, we have a set of scenarios which give different orderings on the decisions in the set. These scenarios could also be interpreted in other ways, for example, in multi-criteria decision making, a scenario would represent a criterion; or in a group decision making context, scenarios would correspond to agents, with their orderings over decisions. Let us look at a small example.

Example 4.1 ► MODS example**.**
Suppose we have a set of alternatives $\mathcal{A} = \{\alpha, \beta, \gamma\}$ and a set of agents $\mathcal{S} = \{A, B\}$, where each agent ranks the alternatives in order of preference. The following table shows these rankings, in descending order of preference.

A	B
α	β
β γ	γ
	α

We can see that for agent A, α is preferred to β, and α is preferred to γ, but β and γ are of equal preference. For agent B, β is preferred to γ, and both β and γ are preferred to α. ▲

4.2.1 Multiple-Ordering Decision Structures

We now give a formal definition of the framework.

Definition 4.1 » Multiple-ordering decision structure
A *multiple-ordering decision structure* (MODS) $\mathcal{G}$ is a tuple $\langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\}\rangle$, where

- $\mathcal{A}$ is a non-empty, finite set of decisions,
- $\mathcal{S}$ is a non-empty (finite or infinite) set of scenarios,
- for each $s \in \mathcal{S}$, relation $\preccurlyeq_s$ is a total preorder on $\mathcal{A}$. «

This gives us a framework where, for a set of decisions $\mathcal{A}$, each scenario s in $\mathcal{S}$ provides an ordering $\preccurlyeq_s$ over $\mathcal{A}$. For each relation $\preccurlyeq_s \in \{\preccurlyeq_s : s \in \mathcal{S}\}$, we also derive the corresponding strict relation $\prec_s$, i.e., for all $\alpha, \beta \in \mathcal{A}$, $\alpha \prec_s \beta$, if and only if, $\alpha \preccurlyeq_s \beta$ and $\beta \not\preccurlyeq_s \alpha$, and the corresponding equivalence relation $\equiv_s$, i.e, for all $\alpha, \beta \in \mathcal{A}$, $\alpha \equiv_s \beta$, if and only if, $\alpha \preccurlyeq_s \beta$ and $\beta \preccurlyeq_s \alpha$ (see Definitions 2.3 and 2.4).

The MODS framework is a more general framework than that described by the multi-aspect decision problem defined in Section 2.3, which we recall here.

Recall » General Decision Making Problem (Definition 2.11)
A *multi-aspect decision problem* is a tuple $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where:

- $\mathcal{A}$ is a finite set of decisions, alternatives or choices,
- $\mathcal{S} = \{1, \ldots, m\}$ is a finite set of decision *aspects*, where each $i \in \mathcal{S}$ labels some preferential aspect of the problem, and for which p_i is a function that specifies the preference value of each decision, i.e., $p_i : \mathcal{A} \to T$,
- T is a scale of preference values, where $\leq$ is a total order on T. «

In a multi-aspect decision problem, we have an ordered scale T, where for each decision under consideration we have a vector of preference values, one for each aspect or scenario. In the MODS framework, we do not assume that there is any such scale T, the assumption is only that in each scenario we have a total ordering on the decisions.

4.2.2 Basic Optimality Notions of a MODS

For any MODS $\mathcal{G} = \langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\}\rangle$, and where α and β are some arbitrary elements (decisions) of $\mathcal{A}$, we define the following preference relations. Firstly we

define the $\preccurlyeq_{\mathrm{N}}$ relation, where a decision necessarily dominates another if it is at least as good as the other in every scenario.

Definition 4.2 » Necessarily dominates
We say that α *necessarily dominates* β, written as $\alpha \preccurlyeq_{\mathrm{N}} \beta$, if and only if

— for all $s \in \mathcal{S}$, $\alpha \preccurlyeq_s \beta$. «

Relation $\preccurlyeq_{\mathrm{N}}$ is the intersection of $\preccurlyeq_s$ over all $s \in \mathcal{S}$, i.e., $\preccurlyeq_{\mathrm{N}} = \bigcap_{s\in\mathcal{S}} \preccurlyeq_s$.

For relation $\preccurlyeq_{\mathrm{N}}$, we define $\prec_{\mathrm{N}}$ as the corresponding strict relation (see Definition 2.3).

Definition 4.3 » Strict-necessarily dominates
We say that α *strict-necessarily dominates* β, written as $\alpha \prec_{\mathrm{N}} \beta$, if and only if

— $\alpha \preccurlyeq_{\mathrm{N}} \beta$ and $\beta \not\preccurlyeq_{\mathrm{N}} \alpha$. «

We also define $\equiv_{\mathrm{N}}$ as the corresponding equivalence relation to $\preccurlyeq_{\mathrm{N}}$ (see Definition 2.4).

Definition 4.4 » Necessarily equivalent
We say that α and β are *necessarily equivalent*, written as $\alpha \equiv_{\mathrm{N}} \beta$, if and only if

— $\alpha \preccurlyeq_{\mathrm{N}} \beta$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. «

Relation $\equiv_{\mathrm{N}}$ is the intersection of $\equiv_s$ over all $s \in \mathcal{S}$, i.e., $\equiv_{\mathrm{N}} = \bigcap_{s\in\mathcal{S}} \equiv_s$.

We now define the $\prec_{\mathrm{NS}}$ relation, where a decision necessarily strictly dominates another if it is strictly better in every scenario.

Definition 4.5 » Necessarily strictly dominates
We say that α *necessarily strictly dominates* β, written $\alpha \prec_{\mathrm{NS}} \beta$, if and only if

— for all $s \in \mathcal{S}$, $\alpha \prec_s \beta$. «

Relation $\prec_{\mathrm{NS}}$ is the intersection of $\prec_s$ over all $s \in \mathcal{S}$, i.e., $\prec_{\mathrm{NS}} = \bigcap_{s\in\mathcal{S}} \prec_s$.

We define, for $\alpha \in \mathcal{A}$, the N-equivalence class of α, which is the subset of all elements in $\mathcal{A}$ that are necessarily equivalent to α (see Definition 2.5).

Definition 4.6 » N-equivalence class
For $\alpha \in \mathcal{A}$, let $[\alpha]_{\mathrm{N}}$ denote the N-*equivalence class* of α, given by

— $[\alpha]_{\mathrm{N}} = \{\beta \in A : \alpha \equiv_{\mathrm{N}} \beta\}$ «

We now define the set of optimal elements and the set of strictly optimal elements with respect to some given scenario.

Definition 4.7 » Optimal in a scenario
For scenario $s \in \mathcal{S}$, let O_s be the set of optimal elements in s, i.e., the set of $\alpha \in \mathcal{A}$ such that for all $\beta \in \mathcal{A}$, $\alpha \preccurlyeq_s \beta$. «

Definition 4.8 » Strictly optimal in a scenario
For scenario $s \in \mathcal{S}$, let SO_s be the set of strictly optimal elements in s, i.e., the set of $\alpha \in \mathcal{A}$ such that $\alpha \prec_s \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$. «

4.2.3 The Basic Optimality Classes

In this section we look at some basic optimality classes which represent different natural notions of optimality in the MODS framework.

Definition 4.9 » Necessarily optimal
We say that α is *necessarily optimal* if and only if

— $\alpha \preccurlyeq_{\mathrm{N}} \beta$, for all $\beta \in \mathcal{A}$. «

That is, α is necessarily optimal if it necessarily dominates every $\beta \in \mathcal{A}$. Let $\mathrm{NO}(\mathcal{G})$ denote the set of necessarily optimal elements for MODS $\mathcal{G}$.

Definition 4.10 » Necessarily strictly optimal
We say that α is *necessarily strictly optimal* if and only if

— $\alpha \prec_{\mathrm{NS}} \beta$, for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$ «

That is, α is necessarily strictly optimal if it necessarily strictly dominates every β that is not equivalent to α. Let $\mathrm{NSO}(\mathcal{G})$ denote the set of necessarily strictly optimal elements for MODS $\mathcal{G}$.

Definition 4.11 » Possibly optimal
We say that α is *possibly optimal* if and only if

- there exists $s \in \mathcal{S}$, such that $\alpha \preccurlyeq_s \beta$ for all $\beta \in \mathcal{A}$. «

That is, α is possibly optimal if it is optimal in some scenario. Let $\text{PO}(\mathcal{G})$ denote the set of possibly optimal elements for MODS $\mathcal{G}$.

Definition 4.12 » Possibly strictly optimal
We say that α is *possibly strictly optimal* if and only if

— there exists $s \in \mathcal{S}$ such that $\alpha \prec_s \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ «

That is, α is possibly strictly optimal if it is strictly optimal in some scenario. Let $\text{PSO}(\mathcal{G})$ denote the set of possibly optimal elements for MODS $\mathcal{G}$.

Definition 4.13 » Can dominate
We say that α *can dominate* any other decision if and only if

— for all $\beta \in \mathcal{A}$ there exists $s \in \mathcal{S}$ such that $\alpha \preccurlyeq_s \beta$. «

That is, α is in $\text{CD}(\mathcal{G})$ if for every decision there exists some scenario in which it dominates that decision. Let $\text{CD}(\mathcal{G})$ denote the set of decisions that can dominate any other decision, for MODS $\mathcal{G}$. We have the following result.

Proposition 4.1 » $\text{CD}(\mathcal{G})$ and $\prec_{\text{NS}}$ result

$\alpha \in \text{CD}(\mathcal{G}) \iff$ *there exists no* $\beta \in \mathcal{A}$ *such that* $\beta \prec_{\text{NS}} \alpha$ ◇

Proof: By definition, we have that $\alpha \in \text{CD}(\mathcal{G})$ if and only if for all $\beta \in \mathcal{A}$ there exists $s \in \mathcal{S}$ such that $\alpha \preccurlyeq_s \beta$, i.e., $\alpha \prec_s \beta$ or $\alpha \equiv_s \beta$. Since $\preccurlyeq_s$ is complete, then $\alpha \in \text{CD}(\mathcal{G})$ if and only if for all $\beta \in \mathcal{A}$ there exists $s \in \mathcal{S}$ such that $\beta \not\prec_s \alpha$. Therefore, $\alpha \in \text{CD}(\mathcal{G})$, if and only if for all $\beta \in \mathcal{A}$, it is not the case that for all $s \in \mathcal{S}$, $\beta \prec_s \alpha$. This gives us that $\alpha \in \text{CD}(\mathcal{G})$, if and only if it is not the case that there exists $\beta \in \mathcal{A}$ such that for all $s \in \mathcal{S}$, $\beta \prec_s \alpha$. By definition of $\prec_{\text{NS}}$, this gives us that $\alpha \in \text{CD}(\mathcal{G})$ if and only if it is not the case that there exists $\beta \in \mathcal{A}$ such that $\beta \prec_{\text{NS}} \alpha$. ∎

Therefore we can see that $\text{CD}(\mathcal{G})$ are the decisions that are undominated with respect to the $\prec_{\text{NS}}$ relation.

Definition 4.14 » Can strictly dominate
We say that α *can strictly dominate* any non-equivalent decision if and only if

— for all $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$, there exists $s \in \mathcal{S}$ such that $\alpha \prec_s \beta$. «

That is, α is in $\text{CSD}(\mathcal{G})$ if for every non-equivalent decision there exists some scenario in which it strictly dominates that decision. Let $\text{CSD}(\mathcal{G})$ denote the set of decisions that can strictly dominate any other decision, for MODS $\mathcal{G}$. We have the following result in relation to $\text{CSD}(\mathcal{G})$.

Proposition 4.2 » $\text{CSD}(\mathcal{G})$ **and** $\prec_{\text{N}}$ **result**

$\alpha \in \text{CSD}(\mathcal{G}) \iff$ *there exists no* $\beta \in \mathcal{A}$ *such that* $\beta \prec_{\text{N}} \alpha$ ◇

Proof: By definition, we have $\alpha \in \text{CSD}(\mathcal{G})$ if and only if for all $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$, there exists $s \in \mathcal{S}$ such that $\alpha \prec_s \beta$. Since $\prec_s$ is the strict part of $\preccurlyeq_s$, and $\preccurlyeq_s$ is complete, then we have that $\alpha \in \text{CSD}(\mathcal{G})$ if and only if for all $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$, there exists $s \in \mathcal{S}$ such that $\beta \not\preccurlyeq_s \alpha$. Therefore, $\alpha \in \text{CSD}(\mathcal{G})$, if and only if for all $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$, it is not the case that for all $s \in \mathcal{S}$, $\beta \preccurlyeq_s \alpha$. This gives us that $\alpha \in \text{CSD}(\mathcal{G})$, if and only if it is not the case that there exists $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ such that for all $s \in \mathcal{S}$, $\beta \preccurlyeq_s \alpha$. By definition of $\preccurlyeq_{\text{N}}$, this gives us that $\alpha \in \text{CSD}(\mathcal{G})$ if and only if it is not the case that there exists $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ such that $\beta \preccurlyeq_{\text{N}} \alpha$, i.e., $\beta \prec_{\text{N}} \alpha$ or $\beta \equiv_{\text{N}} \alpha$. Since $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ does not contain any elements that are equivalent to α, then we have that $\alpha \in \text{CSD}(\mathcal{G})$ if and only if it is not the case that there exists $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ such that $\beta \prec_{\text{N}} \alpha$. ∎

Therefore we can see that $\text{CSD}(\mathcal{G})$ are the decisions that are undominated with respect to $\prec_{\text{N}}$.

We now look at some classes which are given by the intersections of previously defined classes. First, for any class $\text{X}(\mathcal{G})$, let $\text{X}'(\mathcal{G}) = \text{X}(\mathcal{G}) \cap \text{CSD}(\mathcal{G})$. Some specific intersection classes that we use are defined as follows.

Definition 4.15 » Intersection classes
For MODS $\mathcal{G}$,

(i) $\text{NOPSO}(\mathcal{G}) = \text{NO}(\mathcal{G}) \cap \text{PSO}(\mathcal{G})$.

(ii) $\text{PO}'(\mathcal{G}) = \text{PO}(\mathcal{G}) \cap \text{CSD}(\mathcal{G})$.

(iii) For some scenario $s \in \mathcal{S}$, $\text{O}'_s = \text{O}_s \cap \text{CSD}(\mathcal{G})$. «

Now we look at some basic results, which give some properties of the basic optimality classes and some of the relationships between these classes, in particular how the sets of optimal and strictly optimal decisions in a particular scenario relate to the basic classes.

Proposition 4.3 » Basic classes result

For any MODS $\mathcal{G} = \langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\}\rangle$*, where* s *is an arbitrary scenario in* $\mathcal{S}$*, we have the following relationships.*

(i) $\text{SO}_s \subseteq \text{O}'_s \subseteq \text{O}_s$.

(ii) O'_s *and* O_s *are always non-empty.*

(iii) SO_s *is non-empty, if and only if, every pair of decisions in* O_s *are necessarily equivalent. If* SO_s *is non-empty then* $\text{SO}_s = \text{O}'_s = \text{O}_s$.

(iv) $\text{NO}(\mathcal{G}) \subseteq \text{O}_s \subseteq \text{PO}(\mathcal{G})$; $\text{NO}(\mathcal{G}) = \bigcap_{s\in\mathcal{S}} \text{O}_s$; $\text{PO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{O}_s$.

(v) $\text{NSO}(\mathcal{G}) \subseteq \text{SO}_s \subseteq \text{PSO}(\mathcal{G})$; $\text{NSO}(\mathcal{G}) = \bigcap_{s\in\mathcal{S}} \text{SO}_s$; $\text{PSO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{SO}_s$.

(vi) $\text{NO}(\mathcal{G}) \subseteq \text{CSD}(\mathcal{G})$.

(vii) $\text{NO}(\mathcal{G}) \subseteq \text{O}'_s \subseteq \text{PO}'(\mathcal{G})$; $\text{NO}(\mathcal{G}) = \bigcap_{s\in\mathcal{S}} \text{O}'_s$; $\text{PO}'(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{O}'_s$. ◇

Proof:

(i) $\text{O}'_s \subseteq \text{O}_s$ follows directly from definition of O'_s.

Now we show that $\text{SO}_s \subseteq \text{O}'_s$. Suppose, there exists $\alpha \in \mathcal{A}$, such that α is in SO_s but not in O'_s. Then we have either $\alpha \notin \text{O}_s$ or $\alpha \notin \text{CSD}(\mathcal{G})$. Since by definition of SO_s, $\alpha \in \text{SO}_s$ implies that $\alpha \in \text{O}_s$, then we must have $\alpha \notin \text{CSD}(\mathcal{G})$, i.e., there exists $\beta \in \mathcal{A} \setminus [\alpha]_{\text{N}}$ such that $\beta \prec_{\text{N}} \alpha$, and therefore $\beta \preccurlyeq_{\text{N}} \alpha$, i.e., for all $s' \in \mathcal{S}$, $\beta \preccurlyeq_{s'} \alpha$. Therefore $\beta \preccurlyeq_s \alpha$, which contradicts $\alpha \in \text{SO}_s$, since

$\beta \not\equiv_N \alpha$. Therefore we have that if $\alpha \in \mathrm{SO}_s$, then $\alpha \in \mathrm{O}'_s$, and thus $\mathrm{SO}_s \subseteq \mathrm{O}'_s$.

(ii) Since relation $\preccurlyeq_s$ is complete and the set of decisions $\mathcal{A}$ is finite, then there necessarily exists an uniformly-dominating decision with respect to relation $\preccurlyeq_s$, and therefore O_s is always non-empty.

Now we show that O'_s is always non-empty. Suppose for all $\alpha \in \mathcal{A}$, if $\alpha \in \mathrm{O}_s$, then $\alpha \notin \mathrm{CSD}(\mathcal{G})$. Let a_1 be some decision in O_s. Then there exists $a_2 \in \mathcal{A}$ such that $a_2 \prec_{\mathrm{N}} a_1$, so we have for all $s' \in \mathcal{S}$, in particular s, $a_2 \preccurlyeq_{s'} a_1$. Therefore we have $a_2 \in \mathrm{O}_s$, and from our supposition we have that $a_2 \notin \mathrm{CSD}(\mathcal{G})$. Since $\mathcal{A}$ is finite, we can continue until we exhaust all decisions in $\mathcal{A}$, and the final decision a_n is an element of O_s. However a_n is undominated with respect to $\prec_{\mathrm{N}}$, and since $\prec_{\mathrm{N}}$ is acyclic, then $a_n \in \mathrm{CSD}(\mathcal{G})$, which is a contradiction. Therefore we have that there exists $\alpha \in \mathcal{A}$ such that $\alpha \in \mathrm{O}_s$ and $\alpha \in \mathrm{CSD}(\mathcal{G})$, i.e., O'_s is always non-empty.

(iii) First we show SO_s is non-empty, if and only if, every pair of decisions in O_s are necessarily equivalent. Suppose every pair of decisions in O_s are necessarily equivalent. Let α be some decision in O_s. Since by definition of O_s, we have that for all $\beta \in \mathcal{A}$, $\alpha \preccurlyeq_s \beta$, which means $\alpha \equiv_s \beta$ or $\alpha \prec_s \beta$. If $\alpha \equiv_s \beta$, then $\beta \in \mathrm{O}_s$, and we have that $\alpha \equiv_{\mathrm{N}} \beta$. Therefore, we have for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$, $\alpha \prec_s \beta$, i.e., $\alpha \in \mathrm{SO}_s$. Now suppose that exists $\alpha, \beta \in \mathrm{O}_s$ such that $\alpha \not\equiv_{\mathrm{N}} \beta$. Therefore, since we have $\alpha \equiv_s \beta$ i.e., $\alpha \not\prec_s \beta$ and $\beta \not\prec_s \alpha$, then by definition of SO_s, we have that both $\alpha, \beta \notin \mathrm{SO}_s$. Since this is true for any pair of non-necessarily equivalent decisions in O_s, therefore we have that SO_s is empty. This shows SO_s is non-empty if and only if every pair of decisions in O_s are necessarily equivalent.

Now we show that if SO_s is non-empty, then $\mathrm{SO}_s = \mathrm{O}'_s = \mathrm{O}_s$. Suppose SO_s is non-empty, i.e., we have some $\beta \in \mathcal{A}$ such that β is in SO_s. Now suppose that there is some $\alpha \in \mathcal{A}$ such that α is in O_s but not in SO_s. Since $\mathrm{SO}_s \subseteq \mathrm{O}_s$, then β is also in O_s, and since we have from (i) $\alpha \equiv_{\mathrm{N}} \beta$, then this contradicts $\alpha \notin \mathrm{SO}_s$. Therefore we have shown that if SO_s is non-empty, then $\mathrm{O}_s \subseteq \mathrm{SO}_s$, and therefore $\mathrm{SO}_s = \mathrm{O}'_s = \mathrm{O}_s$.

(iv) $\mathrm{NO}(\mathcal{G}) = \bigcap_{s \in \mathcal{S}} \mathrm{O}_s$ follows from definitions of $\preccurlyeq_{\mathrm{N}}$, $\mathrm{NO}(\mathcal{G})$ and O_s.

$\mathrm{PO}(\mathcal{G}) = \bigcup_{s \in \mathcal{S}} \mathrm{O}_s$ follows from definitions of $\mathrm{PO}(\mathcal{G})$ and O_s.

$\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{O}_s \subseteq \mathrm{PO}(\mathcal{G})$ follows from $\mathrm{NO}(\mathcal{G}) = \bigcap_{s \in \mathcal{S}} \mathrm{O}_s$ and $\mathrm{PO}(\mathcal{G}) = \bigcup_{s \in \mathcal{S}} \mathrm{O}_s$.

(v) $\mathrm{NSO}(\mathcal{G}) = \bigcap_{s \in \mathcal{S}} \mathrm{SO}_s$ follows from definitions of $\prec_{\mathrm{NS}}$, $\mathrm{NSO}(\mathcal{G})$ and SO_s.

$\mathrm{PSO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \mathrm{SO}_s$ follows from definitions of $\mathrm{PSO}(\mathcal{G})$ and SO_s.

$\mathrm{NSO}(\mathcal{G}) \subseteq \mathrm{SO}_s \subseteq \mathrm{PSO}(\mathcal{G})$ follows from $\mathrm{NSO}(\mathcal{G}) = \bigcap_{s\in\mathcal{S}} \mathrm{SO}_s$ and $\mathrm{PSO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \mathrm{SO}_s$.

(vi) Suppose there exists $\alpha \in \mathrm{NO}(\mathcal{G})$ such that $\alpha \notin \mathrm{CSD}(\mathcal{G})$. Then there exists $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$ such that $\beta \prec_{\mathrm{N}} \alpha$. This contradicts $\alpha \in \mathrm{NO}(\mathcal{G})$, since for all $\beta \in \mathcal{A}$, $\alpha \preccurlyeq_{\mathrm{N}} \beta$. Therefore, we have $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{CSD}(\mathcal{G})$.

(vii) This follows from definitions of O'_s and $\mathrm{PO}'(\mathcal{G})$, and from (iv) and (vi). ▪

Let us look at an example considering the six basic optimality classes defined to this point.

Table 4.1: An example of seven different scenarios and their associated orderings over a set of decisions $\{\alpha, \beta, \gamma, \delta\}$

$\preccurlyeq_{s_1}$	$\preccurlyeq_{s_2}$	$\preccurlyeq_{s_3}$	$\preccurlyeq_{s_4}$	$\preccurlyeq_{s_5}$	$\preccurlyeq_{s_6}$	$\preccurlyeq_{s_7}$
$\alpha\ \beta$	$\alpha\ \beta$	δ	$\alpha\ \gamma$	$\alpha\ \beta\ \delta$	γ	γ
γ	δ	γ	β	γ	β	δ
δ	γ	β	δ		α	α
		α			δ	β

Example 4.2 ► Basic classes example**.**
Consider MODS $\mathcal{G} = \mathcal{G}_{123}$*, with set of decisions* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$*, set of scenarios* $\mathcal{S} = \{s_1, s_2, s_3\}$*, where the associated total pre-orders are given by Table 4.1.*

For example, the scenario s_1 *has associated total pre-order* $\preccurlyeq_{s_1}$ *over* $\mathcal{A}$ *which is given by the transitive closure of* $\alpha \equiv_{s_1} \beta \prec_{s_1} \gamma \prec_{s_1} \delta$.

We look at the resulting basic optimality classes.

For scenario s_1*, the optimal decisions according to* $\preccurlyeq_{s_1}$ *are* α *and* β*, so we have:*

- $\mathrm{O}_{s_1} = \{\alpha, \beta\}$

For scenario s_3*, with associated total preorder* $\preccurlyeq_{s_3}$*, we have that* δ *is strictly optimal, so we have:*

- $\mathrm{SO}_{s_3} = \{\delta\}$

Since α *and* β *are optimal in the first two scenarios, and* δ *is optimal in the third, and since these are the decisions that are optimal in some scenario, we have:*

- $\mathrm{PO}(\mathcal{G}) = \{\alpha, \beta, \delta\}$

Because no decision is optimal in all scenarios, there is no necessarily optimal decision, so we have:

- $\mathrm{NO}(\mathcal{G}) = \mathrm{NSO}(\mathcal{G}) = \emptyset$

Since δ is strictly optimal in the third scenario, then we have:

- $\mathrm{PSO}(\mathcal{G}) = \{\delta\}$

We have that $\alpha \notin \mathrm{CSD}(\mathcal{G})$, since in none of the three scenarios does it strictly dominate β, and since all other decisions can strictly dominate any other decision, we have:

- $\mathrm{CSD}(\mathcal{G}) = \{\beta, \gamma, \delta\}$

Since $\mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G}) \cap \mathrm{PSO}(\mathcal{G})$, and $\mathrm{NO}(\mathcal{G})$ is empty, then we have:

- $\mathrm{NOPSO}(\mathcal{G}) = \emptyset$

Since $\mathrm{PO}'(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) \cap \mathrm{CSD}(\mathcal{G})$, then we have:

- $\mathrm{PO}'(\mathcal{G}) = \{\beta, \delta\}$

Finally, since no decision is strictly dominated by another in all scenarios, we have:

- $\mathrm{CD}(\mathcal{G}) = \{\alpha, \beta, \gamma, \delta\} = \mathcal{A}$ ▲

4.2.4 Relations between Basic Classes

In this section, we look at the relationships between the basic optimality classes, examining which classes are subsets of other classes, and also which classes are always non-empty.

Firstly, we define the following property for *closed under improvement*, and give a basic result in relation to it.

Definition 4.16 » Closed under improvement
For some $\mathrm{X} \subseteq \mathcal{A}$, we say that X is *closed under improvement* if the following property holds:

— for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \in \mathrm{X}$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$ then $\beta \in \mathrm{X}$. «

Proposition 4.4 » Closed under improvement result

For MODS $\mathcal{G}$*, let* X *be a subset of* $\mathcal{A}$ *that is closed under improvement.*

(i) $\mathrm{NO}(\mathcal{G}) \cap \mathrm{X} \neq \emptyset$ *implies* $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{X}$*; hence, if also* $\mathrm{X} \subseteq \mathrm{NO}(\mathcal{G})$ *then* $\mathrm{X} = \mathrm{NO}(\mathcal{G})$.

(ii) $\mathrm{X} \neq \emptyset$ *implies* $\mathrm{X} \cap \mathrm{CSD}(\mathcal{G}) \neq \emptyset$.

(iii) $\mathrm{NSO}(\mathcal{G})$, $\mathrm{NOPSO}(\mathcal{G})$, $\mathrm{PSO}(\mathcal{G})$, $\mathrm{NO}(\mathcal{G})$, $\mathrm{PO}'(\mathcal{G})$, $\mathrm{PO}(\mathcal{G})$, $\mathrm{CSD}(\mathcal{G})$, $\mathrm{CD}(\mathcal{G})$, SO_s *and* O_s *are closed under improvement.* ◇

Proof:

(i) Suppose $\mathrm{NO}(\mathcal{G}) \cap \mathrm{X} \neq \emptyset$, i.e., there exists $\alpha \in \mathrm{NO}(\mathcal{G}) \cap \mathrm{X}$. Now suppose there exists $\beta \in \mathrm{NO}(\mathcal{G})$ such that $\beta \notin \mathrm{X}$. Since both $\alpha, \beta \in \mathrm{NO}(\mathcal{G})$, then we have that $\alpha \equiv_{\mathrm{N}} \beta$, and since X is closed under improvement, then we have $\beta \in \mathrm{X}$. This is a contradiction, so we have that $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{X}$.

Also, if $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{X}$, then since $\mathrm{X} \subseteq \mathrm{NO}(\mathcal{G})$, we obviously have $\mathrm{NO}(\mathcal{G}) = \mathrm{X}$.

(ii) Suppose $\mathrm{X} \neq \emptyset$. Now suppose $\mathrm{X} \cap \mathrm{CSD}(\mathcal{G}) = \emptyset$. Then there is no decision $\alpha \in \mathrm{X}$ such that $\alpha \in \mathrm{CSD}(\mathcal{G})$, i.e., for all $\alpha \in \mathrm{X}$, there exists some $\beta \in \mathcal{A}$ such that $\beta \prec_{\mathrm{N}} \alpha$. Let a_1 be some decision in X. Then there exists $a_2 \in \mathcal{A}$ such that $a_2 \prec_{\mathrm{N}} a_1$. Since X is closed under improvement, then $a_2 \in \mathrm{X}$, which means that there exists some $a_3 \in \mathcal{A}$ such that $a_3 \prec_{\mathrm{N}} a_2$. Since $\mathcal{A}$ is finite, we can continue like this until we exhaust all elements of $\mathcal{A}$, and for the final element a_n there is no remaining element $a_{n+1} \in \mathcal{A}$ such that $a_{n+1} \prec_{\mathrm{N}} a_n$. We have $a_n \in \mathrm{X}$, and since a_n is undominated with respect to $\prec_{\mathrm{N}}$, then $a_n \in \mathrm{CSD}(\mathcal{G})$, which is a contradiction. Therefore we have that $\mathrm{X} \cap \mathrm{CSD}(\mathcal{G})$ is non-empty.

(iii) By definition, $\mathrm{NO}(\mathcal{G})$ is closed under improvement.

Now we show $\mathrm{NSO}(\mathcal{G})$ is closed under improvement. Suppose that $\alpha \in \mathrm{NSO}(\mathcal{G})$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. Now suppose that $\beta \notin \mathrm{NSO}(\mathcal{G})$, i.e, there exists $\alpha' \in \mathcal{A} \setminus [\beta]_{\mathrm{N}}$ such that $\beta \not\prec_{\mathrm{NS}} \alpha'$. However since $\beta \preccurlyeq_{\mathrm{N}} \alpha$, and $\alpha \prec_{\mathrm{NS}} \alpha'$ for all $\alpha' \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$, then this leads to a contradiction. Therefore we have $\beta \in \mathrm{NSO}(\mathcal{G})$, and thus $\mathrm{NSO}(\mathcal{G})$ is closed under improvement.

Here, we show $\mathrm{PSO}(\mathcal{G})$ is closed under improvement. Suppose that $\alpha \in \mathrm{PSO}(\mathcal{G})$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. Since there exists $s' \in \mathcal{S}$ such that $\alpha \prec_{s'} \alpha'$, for all $\alpha' \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$, and since for all $s \in \mathcal{S}$, in particular s', $\beta \preccurlyeq_s \alpha$, then β must be strictly

optimal in scenario s'. Therefore $\beta \in \mathrm{PSO}(\mathcal{G})$ and thus $\mathrm{PSO}(\mathcal{G})$ is closed under improvement.

A proof similar to that used for $\mathrm{PSO}(\mathcal{G})$ can also be used to prove $\mathrm{PO}(\mathcal{G})$ is closed under improvement.

We show $\mathrm{CD}(\mathcal{G})$ is closed under improvement. Suppose that $\alpha \in \mathrm{CD}(\mathcal{G})$, and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. Since for all $\alpha' \in \mathcal{A}$, there exists a scenario s' such that $\alpha \preccurlyeq_{s'} \alpha'$, and since for all $s \in \mathcal{S}$, in particular s', $\beta \preccurlyeq_s \alpha$, then we have that $\beta \in \mathrm{CD}(\mathcal{G})$. Therefore $\mathrm{CD}(\mathcal{G})$ is closed under improvement.

We prove $\mathrm{CSD}(\mathcal{G})$ is closed under improvement as follows. Suppose that $\alpha \in \mathrm{CSD}(\mathcal{G})$, and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. Now suppose that $\beta \notin \mathrm{CSD}(\mathcal{G})$. Then there exists some $\alpha' \in \mathcal{A} \setminus [\beta]_{\mathrm{N}}$ such that $\alpha' \prec_{\mathrm{N}} \beta$. However since $\alpha \in \mathrm{CSD}(\mathcal{G})$ we have for all $\alpha' \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$ that $\alpha' \not\prec_{\mathrm{N}} \alpha$, and since $\beta \preccurlyeq_{\mathrm{N}} \alpha$, then this leads to a contradiction. Therefore $\beta \in \mathrm{CSD}(\mathcal{G})$ and thus $\mathrm{CSD}(\mathcal{G})$ is closed under improvement.

SO_s is closed under improvement, since if α is in SO_s and $\beta \preccurlyeq_{\mathrm{N}} \alpha$, then for all $s' \in \mathcal{S}$, in particular s, $\beta \preccurlyeq_{s'} \alpha$, and therefore $\beta \in \mathrm{SO}_s$.

By a similar argument used for SO_s, O_s is closed under improvement. ∎

The following result gives the basic subset relationships between the classes we have introduced to this point. We use the notation $A \subseteq (B, C) \subseteq D$ to mean that $A \subseteq B \subseteq D$ and $A \subseteq C \subseteq D$.

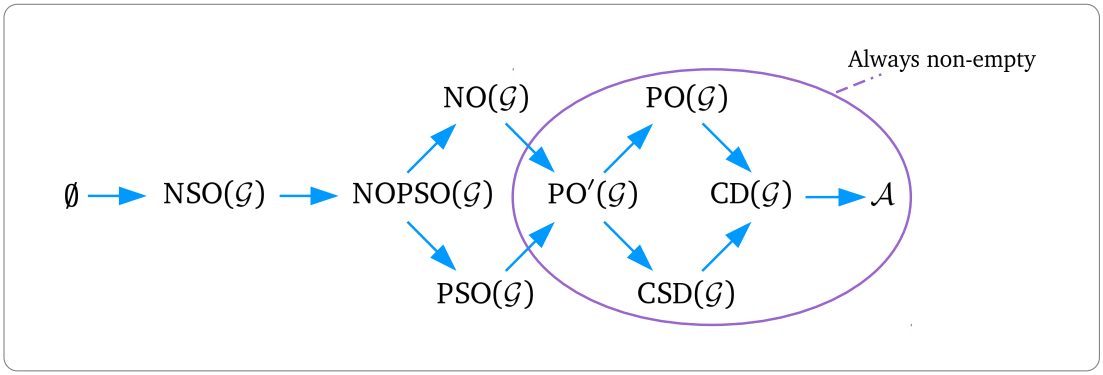


Figure 4.1: Subclass relationships for the basic optimality classes.

Proposition 4.5 » Basic classes hierarchy

For all MODS $\mathcal{G}$*,*

(i) The classes satisfy the following relationships:

$$\text{NSO}(\mathcal{G}) \subseteq \text{NOPSO} \subseteq (\text{NO}(\mathcal{G}), \text{PSO}(\mathcal{G})) \subseteq \text{PO}'(\mathcal{G}) \subseteq (\text{PO}(\mathcal{G}), \text{CSD}(\mathcal{G})) \subseteq \text{CD}(\mathcal{G}).$$

(ii) $\text{PO}'(\mathcal{G})$*,* $\text{PO}(\mathcal{G})$*,* $\text{CSD}(\mathcal{G})$ *and* $\text{CD}(\mathcal{G})$ *are always non-empty.* ◇

Proof:

(i) $\text{NSO}(\mathcal{G}) \subseteq \text{NOPSO} \subseteq (\text{NO}(\mathcal{G}), \text{PSO}(\mathcal{G}))$ follows from the definitions of the classes, as does $\text{PO}'(\mathcal{G}) \subseteq (\text{PO}(\mathcal{G}), \text{CSD}(\mathcal{G}))$ and $\text{CSD}(\mathcal{G}) \subseteq \text{CD}(\mathcal{G})$.

$\text{NO}(\mathcal{G}) \subseteq \text{PO}'(\mathcal{G})$ follows directly from (vii) in Proposition 4.3.

$\text{PSO}(\mathcal{G}) \subseteq \text{PO}'(\mathcal{G})$ follows from (i), (v) and (vii) of Proposition 4.3, i.e., $\text{SO}_s \subseteq \text{O}'_s$, $\text{PSO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{SO}_s$, and $\text{PO}'(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{O}'_s$.

Finally we show $\text{PO}(\mathcal{G}) \subseteq \text{CD}(\mathcal{G})$. From Proposition 4.3 (iv), we have $\text{PO}(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{O}_s$. Since for any $s \in \mathcal{S}$, we have for some $\alpha \in \mathcal{A}$ that $\alpha \in \text{O}_s$, if and only if, $\alpha \preccurlyeq_s \beta$ for all $\beta \in \mathcal{A}$, then by definition of $\text{CD}(\mathcal{G})$, $\alpha \in \text{CD}(\mathcal{G})$, and thus $\text{PO}(\mathcal{G}) \subseteq \text{CD}(\mathcal{G})$.

(ii) $\text{PO}'(\mathcal{G})$ is non-empty since $\text{PO}'(\mathcal{G}) = \bigcup_{s\in\mathcal{S}} \text{O}'_s$ and from Proposition 4.3 (ii), we have, for any $s \in \mathcal{S}$, O'_s is non-empty.

Since $\text{PO}'(\mathcal{G})$ is non-empty, then it follows from the class hierarchy given in (i) that $\text{PO}(\mathcal{G})$, $\text{CSD}(\mathcal{G})$ and $\text{CD}(\mathcal{G})$ are non-empty. ∎

4.2.5 Discussion of the Basic Classes

In this section, we look at the basic classes and discuss some of the relationships between these classes, in the context of our decision-making task for selecting a subset of decisions in the MODS framework.

Necessarily optimal

The class NO($\mathcal{G}$) consists of all the decisions that are optimal in every scenario. NO($\mathcal{G}$) will often be empty, but if it is not, then the subclass graph collapses into a chain, as we will see in Proposition 4.8.

Necessarily strictly optimal

The class NSO($\mathcal{G}$) consists of the decisions that are strictly optimal in every scenario, i.e., when all scenarios agree that exactly the same decisions are optimal. NSO($\mathcal{G}$) will nearly always be empty, but if it is not, then all the classes collapse into one class, as we will see in Proposition 4.8.

Possibly optimal

The class PO($\mathcal{G}$) consists the decisions that are optimal in some scenario. For decision making under uncertainty, where we consider that some scenario ordering $\preccurlyeq_s$ gives the correct ordering on $\mathcal{A}$, the possibly optimal decisions are the ones that could be the best.

Can dominate

The class CD($\mathcal{G}$) contains the decisions that are undominated with respect to $\prec_{\text{NS}}$, so if a decision α is not in CD($\mathcal{G}$), then there exists some decision that is strictly better in every scenario. Thus, being a member of CD($\mathcal{G}$) is a very weak notion of optimality, and it would be hard to argue that any decision that is not in CD($\mathcal{G}$) should be viewed as being an optimal decision.

Can strictly dominate

The class CSD($\mathcal{G}$) contains the decisions that are undominated with respect to $\prec_{\text{N}}$, so if a decision α is not in CSD($\mathcal{G}$), then there exists some decision that is at least as good in every scenario, and better in some scenario. Because of this, we argue that being a member of CSD($\mathcal{G}$) is a minimal requirement for a notion of "optimality".

From Proposition 4.4, if X is non-empty and closed under improvement, then $X' = X \cap CSD(\mathcal{G})$ is non-empty, hence we can reduce any such non-empty class X to the potentially smaller non-empty set X′, by eliminating decisions not in CSD($\mathcal{G}$).

4.2.6 Maximally Possibly Optimal Decisions

In this section, we define another optimality class, which is a refinement to the notion of *possibly optimal*. First, recall that O_s is the set of optimal decisions in s, i.e., the set of $\alpha \in \mathcal{A}$ such that $\alpha \preccurlyeq_s \beta$ for all $\beta \in \mathcal{A}$. For each $\alpha \in \mathcal{A}$, we define Opt(α) to consist of the set of scenarios in which α is optimal, i.e., $Opt(\alpha) = \{s \in \mathcal{S} : \alpha \in O_s\}$. Now, let $Opt(\mathcal{A}) = \{Opt(\alpha) : \alpha \in \mathcal{A}\}$. We define the notion of maximally possibly optimal as follows:

Definition 4.17 » Maximally possibly optimal
We say that α is *maximally possibly optimal*, if and only if

— Opt(α) is a maximal subset (w.r.t. $\subseteq$) in Opt($\mathcal{A}$). «

That is, α is maximally possibly optimal if α is optimal in a maximal set of scenarios. We let MPO($\mathcal{G}$) denote the set of maximally possibly optimal decisions. From the definition, we can see that if we have a decision α that is not in MPO($\mathcal{G}$), then there exists some β such that β is optimal in more scenarios than α.

We also explicitly define the intersection of MPO($\mathcal{G}$) and CSD($\mathcal{G}$) as follows.

Definition 4.18 » MPO′($\mathcal{G}$) definition
For MODS $\mathcal{G}$, let $MPO'(\mathcal{G}) = MPO(\mathcal{G}) \cap CSD(\mathcal{G})$. «

Let us look at an example for maximally possibly optimal.

Example 4.3 ► MPO($\mathcal{G}$) example**.**
Let us consider MODS $\mathcal{G} = \mathcal{G}_{15}$*, where we have set of decisions* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$*, set of scenarios* $\mathcal{S} = \{s_1, s_5\}$*, and with associated total pre-orders given in Table 4.1, which we recall here:*

- $\preccurlyeq_{s_1}$ *is the transitive closure of* $\alpha \equiv_{s_1} \beta \prec_{s_1} \gamma \prec_{s_1} \delta$.
- $\preccurlyeq_{s_5}$ *is the transitive closure of* $\alpha \equiv_{s_5} \beta \equiv_{s_5} \delta \prec_{s_5} \gamma$.

We can see that the set of scenarios for which each decision is optimal is as follows:

- $\mathrm{Opt}(\alpha) = \{s_1, s_5\}$
- $\mathrm{Opt}(\beta) = \{s_1, s_5\}$
- $\mathrm{Opt}(\gamma) = \emptyset$
- $\mathrm{Opt}(\delta) = \{s_5\}$

The maximal sets of scenarios are given by:

- $\max \mathrm{Opt}(\mathcal{A}) = \max\{\{s_1, s_5\}, \{s_5\}, \emptyset\} = \{\{s_1, s_5\}\}$

Therefore the decisions that are optimal in a maximal subset of scenarios, i.e., the maximally possibly optimal decisions, are:

- $\mathrm{MPO}(\mathcal{G}) = \{\alpha, \beta\}$ ▲

For $\mathrm{MPO}(\mathcal{G})$ and $\mathrm{MPO}'(\mathcal{G})$, we have the following result.

Proposition 4.6 » $\mathrm{MPO}(\mathcal{G})$ **result**

For all MODS $\mathcal{G}$*,*

(i) $\mathrm{MPO}(\mathcal{G})$ *is non-empty.*

(ii) $\mathrm{MPO}(\mathcal{G})$ *is closed under improvement.*

(iii) $\mathrm{NO}(\mathcal{G}), \mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{MPO}(\mathcal{G}) \subseteq \mathrm{PO}(\mathcal{G})$.

(iv) $\mathrm{NO}(\mathcal{G}), \mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{MPO}'(\mathcal{G}) \subseteq \mathrm{PO}'(\mathcal{G})$. ◇

Proof:

(i) Since O_s is non-empty for any $s \in \mathcal{S}$, then there exists some $\alpha \in \mathcal{A}$ such that $\mathrm{Opt}(\alpha) \supseteq \{s\}$. Therefore $\mathrm{Opt}(\mathcal{A})$ is non-empty, so there exists a maximal subset in $\mathrm{Opt}(\mathcal{A})$, i.e., $\max \mathrm{Opt}(\mathcal{A})$ is non-empty. This means that there exists some α for which $\mathrm{Opt}(\alpha) \in \max \mathrm{Opt}(\mathcal{A})$, i.e., it is maximally possibly optimal, so we have that $\mathrm{MPO}(\mathcal{G})$ is non empty.

(ii) Suppose that α is in $\mathrm{MPO}(\mathcal{G})$, and $\beta \preccurlyeq_{\mathrm{N}} \alpha$. Since $\mathrm{Opt}(\alpha) \in \max \mathrm{Opt}(\mathcal{A})$, and since for all $s \in \mathcal{S}$, $\beta \preccurlyeq_s \alpha$, then we must have $\mathrm{Opt}(\beta) \supseteq \mathrm{Opt}(\alpha)$, therefore $\mathrm{Opt}(\beta) \in \max \mathrm{Opt}(\mathcal{A})$, and thus $\beta \in \mathrm{MPO}(\mathcal{G})$. Therefore $\mathrm{MPO}(\mathcal{G})$ is closed under improvement.

(iii) By definitions of $\mathrm{MPO}(\mathcal{G})$ and $\mathrm{PO}(\mathcal{G})$, we have $\mathrm{MPO}(\mathcal{G}) \subseteq \mathrm{PO}(\mathcal{G})$, since $\mathrm{MPO}(\mathcal{G})$ consists of the decisions that are optimal in a maximal set of scenarios, and $\mathrm{PO}(\mathcal{G})$ consists of the decisions that are optimal in any scenario.

By definitions of $\mathrm{NO}(\mathcal{G})$ and $\mathrm{MPO}(\mathcal{G})$, we have $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{MPO}(\mathcal{G})$, since $\mathrm{NO}(\mathcal{G})$ consists of the decisions which are optimal in all scenarios, which is obviously a maximal set of scenarios.

Now we show that $\mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{MPO}(\mathcal{G})$. Suppose there is some $\alpha \in \mathrm{PSO}(\mathcal{G})$, i.e., there exists some scenario $s \in \mathcal{S}$ such that $\alpha \prec_s \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$. Now suppose that α is not in $\mathrm{MPO}(\mathcal{G})$. Then we have that $\mathrm{Opt}(\alpha)$, which contains s, is not in $\max \mathrm{Opt}(\mathcal{A})$. However, by definition of $\mathrm{PSO}(\mathcal{G})$, since $\alpha \prec_s \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$, then there can be no other $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$ such that $s \in \mathrm{Opt}(\beta)$, so $\mathrm{Opt}(\alpha)$ must be in $\max \mathrm{Opt}(\mathcal{A})$, which is a contradiction. Therefore we have that $\mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{MPO}(\mathcal{G})$.

(iv) $\mathrm{MPO}'(\mathcal{G}) \subseteq \mathrm{PO}'(\mathcal{G})$ follows from (iii).

$\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{MPO}'(\mathcal{G})$ follows from (iii) and from Proposition 4.3 (vi), i.e, $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{CSD}(\mathcal{G})$.

$\mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{MPO}'(\mathcal{G})$ follows from (iii) and from Proposition 4.5 (i), i.e., $\mathrm{PSO}(\mathcal{G}) \subseteq \mathrm{CSD}(\mathcal{G})$. ∎

4.2.7 Extreme Decisions

In this section, we define another refinement of *possibly optimal*, based on the notion of extreme solution in multi-criteria optimisation [Jun04]. For this notion of optimality, we consider the optimal decisions with respect to some scenario s_1, and then (since in this context we are minimising) we consider the minimal decisions of this set with respect to another scenario s_2; this is continued until the set of decisions are all equivalent, and these are the extreme decisions. To aid the definition of an extreme decision, first we define the following.

Let $s_1, \ldots, s_k$ be a finite sequence of scenarios. Let $A_{s_1} = \min_{\preccurlyeq_{s_1}} \mathcal{A} = \mathrm{O}_{s_1}$. For $i = 2, \ldots, k$, let $A_{s_1,\ldots,s_i}$ be $\min \preccurlyeq_{s_i} A_{s_1,\ldots,s_{i-1}}$. Let us call a sequence $s_1, \ldots, s_k$ a *sufficient sequence* if all the elements of $A_{s_1,\ldots,s_k}$ are necessarily equivalent.

Definition 4.19 » Extreme decisions
We say that α is an *extreme decision*, if and only if

— there exists a sufficient sequence $s_1, \ldots, s_k$ of scenarios such that $\alpha \in A_{s_1,\ldots,s_k}$ «

We let EXT($\mathcal{G}$) denote the set of extreme decisions. We also define the intersection of MPO($\mathcal{G}$) and EXT($\mathcal{G}$).

Definition 4.20 » MPO-EXT($\mathcal{G}$) **definition**
For MODS $\mathcal{G}$, let MPO-EXT($\mathcal{G}$) = MPO($\mathcal{G}$) ∩ EXT($\mathcal{G}$). «

Let us now look at an example with EXT($\mathcal{G}$).

Example 4.4 ► EXT($\mathcal{G}$) example**.**
Let us consider MODS $\mathcal{G} = \mathcal{G}_{345}$*, where we have set of decisions* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$*, set of scenarios* $\mathcal{S} = \{s_3, s_4, s_5\}$*, and with associated total pre-orders given in Table 4.1, which we recall here:*

- $\preccurlyeq_{s_3}$ *is the transitive closure of* $\delta \prec_{s_3} \gamma \prec_{s_3} \beta \prec_{s_3} \alpha$.
- $\preccurlyeq_{s_4}$ *is the transitive closure of* $\alpha \equiv_{s_4} \gamma \prec_{s_3} \beta \prec_{s_3} \delta$.
- $\preccurlyeq_{s_5}$ *is the transitive closure of* $\alpha \equiv_{s_5} \beta \equiv_{s_5} \delta \prec_{s_5} \gamma$.

We can see that:

- $\alpha \in$ EXT($\mathcal{G}$) *since for sufficient sequence* s_4, s_5 *we have* $\alpha \in A_{s_4,s_5}$*, which is the set* $\{\alpha\}$.
- $\gamma \in$ EXT($\mathcal{G}$) *since for sufficient sequence* s_4, s_3 *we have* $\gamma \in A_{s_4,s_3}$*, which is the set* $\{\gamma\}$.
- $\delta \in$ EXT($\mathcal{G}$) *since for sufficient sequence* s_3 *we have* $\gamma \in A_{s_3}$*, which is the set* $\{\delta\}$.
- $\beta \notin$ EXT($\mathcal{G}$) *since there is no sufficient sequence* $s_1, \ldots, s_k$ *such that* $A_{s_1,\ldots,s_k}$ *contains* β.

Therefore we have the following set of extreme decisions

- EXT($\mathcal{G}$) = $\{\alpha, \gamma, \delta\}$. ▲

We now give the following results in relation to EXT($\mathcal{G}$) and MPO-EXT($\mathcal{G}$):

Proposition 4.7 » EXT($\mathcal{G}$) result

For all MODS $\mathcal{G}$,

(i) EXT($\mathcal{G}$) *is always non-empty.*

(ii) EXT($\mathcal{G}$) *is closed under improvement.*

(iii) $(\text{NO}(\mathcal{G}), \text{PSO}(\mathcal{G})) \subseteq \text{EXT}(\mathcal{G}) \subseteq (\text{PO}(\mathcal{G}), \text{CSD}(\mathcal{G}))$*, so* $\text{EXT}(\mathcal{G}) \subseteq \text{PO}'(\mathcal{G})$.

(iv) Furthermore, MPO-EXT($\mathcal{G}$) *is always non-empty.* ◇

Proof:

(i) This follows from (iv), since MPO-EXT($\mathcal{G}$) $\subseteq$ EXT($\mathcal{G}$) and MPO-EXT($\mathcal{G}$) is non-empty.

(ii) We can show, by induction on i, that each $A_{s_1,\ldots,s_i}$ is closed under improvement. This also implies that EXT($\mathcal{G}$) is closed under improvement.

(iii) First, we show that EXT($\mathcal{G}$) $\subseteq$ PO($\mathcal{G}$). Consider any $\alpha \in \text{EXT}(\mathcal{G})$, so there exists a sufficient sequence $s_1, \ldots, s_k$ with $A_{s_1,\ldots,s_k}$ containing α. $A_{s_1,\ldots,s_k}$ is a subset of A_{s_1} which equals O_{s_1}, which is a subset of PO($\mathcal{G}$). Thus $\alpha \in \text{PO}(\mathcal{G})$, and therefore EXT($\mathcal{G}$) $\subseteq$ PO($\mathcal{G}$).

Now we show that EXT($\mathcal{G}$) $\subseteq$ CSD($\mathcal{G}$). Suppose that $\alpha \notin \text{CSD}(\mathcal{G})$. Now suppose that $\alpha \in \text{EXT}(\mathcal{G})$, so that there exists a sufficient sequence $s_1, \ldots, s_k$ such that $\alpha \in A_{s_1,\ldots,s_k}$. Since $\alpha \notin \text{CSD}(\mathcal{G})$, then there exists β such that $\beta \prec_{\text{N}} \alpha$. Since $\alpha \in A_{s_1,\ldots,s_k}$, and $\beta \preccurlyeq_{\text{N}} \alpha$, we have $\beta \in A_{s_1,\ldots,s_k}$ because from (ii) we have that $A_{s_1,\ldots,s_k}$ is closed under improvement. This implies that β and α are equivalent, which contradicts the fact that $\beta \prec_{\text{N}} \alpha$. Therefore $\alpha \notin \text{EXT}(\mathcal{G})$, and thus EXT($\mathcal{G}$) $\subseteq$ CSD($\mathcal{G}$).

Here we show NO($\mathcal{G}$) $\subseteq$ EXT($\mathcal{G}$). Suppose $\alpha \in \text{NO}(\mathcal{G})$. Consider any sufficient sequence $s_1, \ldots, s_k$ and let β be some element of $A_{s_1,\ldots,s_k}$. Since $\alpha \in \text{NO}(\mathcal{G})$, then $\alpha \preccurlyeq_{\text{N}} \beta$ and, since $A_{s_1,\ldots,s_k}$ is closed under improvement, $\alpha \in A_{s_1,\ldots,s_k}$, and hence $\alpha \in \text{EXT}(\mathcal{G})$. Therefore we have NO($\mathcal{G}$) $\subseteq$ EXT($\mathcal{G}$).

Finally, we show PSO($\mathcal{G}$) $\subseteq$ EXT($\mathcal{G}$). Suppose $\alpha \in \text{PSO}(\mathcal{G})$, so there exists some scenario s with $\alpha \in \text{SO}_s$. Then from Proposition 4.5 ((iii)), we have $\text{O}_s = \text{SO}_s$, and the elements of O_s are equivalent to each other. Thus s (on its own) is a sufficient sequence. $\alpha \in A_s = \text{O}_s$ implies that $\alpha \in \text{EXT}(\mathcal{G})$, and

therefore $\text{PSO}(\mathcal{G}) \subseteq \text{EXT}(\mathcal{G})$.

(iv) By Proposition 4.6 (i), $\text{MPO}(\mathcal{G})$ is non-empty, so we can choose some decision $\alpha \in \text{MPO}(\mathcal{G})$. Consider $B = \bigcap_{s \in \text{Opt}(\alpha)} \text{O}_s$, which contains α. Since $\mathcal{A}$ is finite, then there exists some finite subset T of $\text{Opt}(\alpha)$ such that $B = \bigcap_{s \in T} \text{O}_s$. Choose $\{s_1, \ldots, s_{|T|}\}$ to be the decisions of T in any order. Then $A_{s_1,\ldots,s_{|T|}} = B$. For each $i = |T|+1, |T|+2, \ldots$ we choose any scenario $s_i \in \mathcal{S}$ such that $A_{s_1,\ldots,s_i}$ is a strict subset of $A_{s_1,\ldots,s_{i-1}}$. If this is not possible then all decisions of $A_{s_1,\ldots,s_{i-1}}$ are equivalent, so we stop. Since $\mathcal{A}$, and hence each $A_{s_1,\ldots,s_i}$, is finite the process must terminate at some point, giving a set $A_{s_1,\ldots,s_k}$ which is non-empty by construction; let β be any decision of it. $\beta \in \text{EXT}(\mathcal{G})$, by definition of $\text{EXT}(\mathcal{G})$. Also, $A_{s_1,\ldots,s_k} \subseteq B$, so $\beta \in B$, and therefore $\text{Opt}(\beta) \supseteq \text{Opt}(\alpha)$. Hence $\text{Opt}(\beta) = \text{Opt}(\alpha)$, since $\alpha \in \text{MPO}(\mathcal{G})$. Thus $\beta \in \text{MPO}(\mathcal{G})$, so $\beta \in \text{MPO-EXT}(\mathcal{G})$, showing that $\text{MPO-EXT}(\mathcal{G})$ is non-empty. ∎

4.3 Subclass Relationships

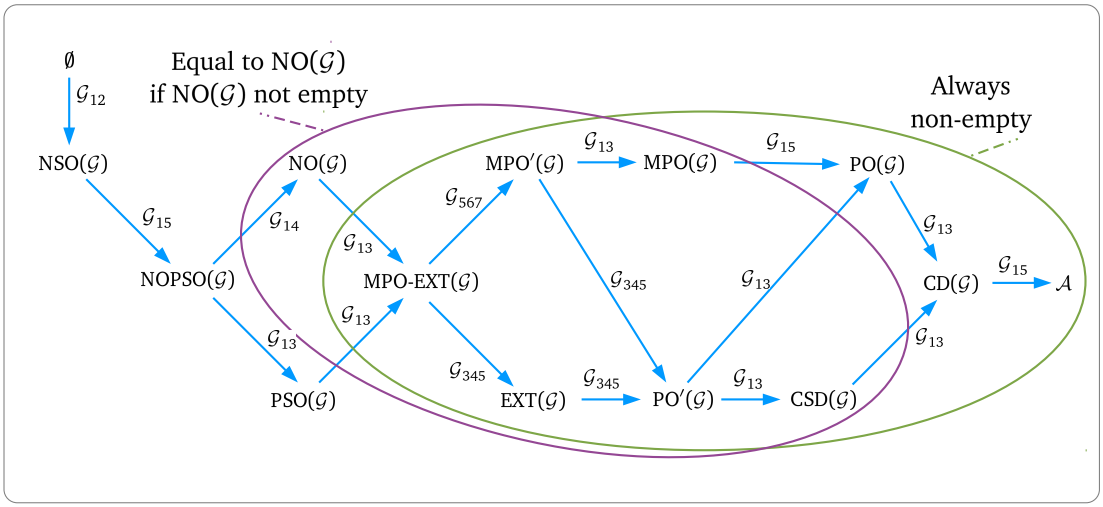


Figure 4.2: Subclass relationships, including examples showing inequality between adjacent optimality classes - see Theorem 4.1 and Section 4.4.1.

In this section, we show the precise subclass relationships between the different definitions of optimality. In the result below, we define, for any $\mathcal{G}$, $\emptyset(\mathcal{G})$ to be $\emptyset$, and $\mathcal{A}(\mathcal{G})$ to be $\mathcal{A}$, the set of decisions in $\mathcal{G}$.

Theorem 4.1 » Subclass relationships for optimality classes

Consider the directed acyclic graph given in Figure 4.2, including edges:
$\emptyset \rightarrow \text{NSO}(\mathcal{G})$, $\text{NSO}(\mathcal{G}) \rightarrow \text{NOPSO}(\mathcal{G})$, *and so on.*
The vertices of the graph are:
$\mathcal{C} = \{\emptyset, \text{NSO}(\mathcal{G}), \text{NOPSO}(\mathcal{G}), \text{NO}(\mathcal{G}), \text{PSO}(\mathcal{G}), \text{MPO-EXT}(\mathcal{G}), \text{EXT}(\mathcal{G}), \text{MPO}'(\mathcal{G}), \text{MPO}(\mathcal{G}), \text{PO}'(\mathcal{G}), \text{PO}(\mathcal{G}), \text{CSD}(\mathcal{G}), \text{CD}(\mathcal{G}), \mathcal{A}\}$.

Let $\mathcal{L}$ be the transitive closure of this graph, so that $(\text{X}, \text{Y}) \in \mathcal{L}$ if and only if there exists a directed path from X *to* Y. *Then for any different $\text{X}, \text{Y} \in \mathcal{C}$*

— $(\text{X}, \text{Y}) \in \mathcal{L}$ *if and only if for all* MODS $\mathcal{G}$, $\text{X}(\mathcal{G}) \subseteq \text{Y}(\mathcal{G})$

Furthermore, $\text{MPO-EXT}(\mathcal{G})$, $\text{EXT}(\mathcal{G})$, $\text{MPO}'(\mathcal{G})$, $\text{MPO}(\mathcal{G})$, $\text{PO}'(\mathcal{G})$, $\text{PO}(\mathcal{G})$, $\text{CSD}(\mathcal{G})$ *and* $\text{CD}(\mathcal{G})$ *are non-empty for any $\mathcal{G}$, but there exists $\mathcal{G}$ such that* $\text{NSO}(\mathcal{G})$, $\text{NOPSO}(\mathcal{G})$, $\text{NO}(\mathcal{G})$ *and* $\text{PSO}(\mathcal{G})$ *are empty.* ◇

Proof: If $(\text{X}, \text{Y}) \in \mathcal{L}$ then by Propositions 4.5, 4.6 and 4.7, for all $\mathcal{G}$, $\text{X}(\mathcal{G}) \subseteq \text{Y}(\mathcal{G})$. These propositions also show that $\text{MPO-EXT}(\mathcal{G})$, $\text{EXT}(\mathcal{G})$, $\text{MPO}'(\mathcal{G})$, $\text{MPO}(\mathcal{G})$, $\text{PO}'(\mathcal{G})$, $\text{PO}(\mathcal{G})$, $\text{CSD}(\mathcal{G})$ and $\text{CD}(\mathcal{G})$ are non-empty for any $\mathcal{G}$.

Conversely, suppose that $(\text{X}, \text{Y}) \notin \mathcal{L}$. We need to prove that there exists some $\mathcal{G}$ such that $\text{X}(\mathcal{G}) \not\subseteq \text{Y}(\mathcal{G})$.

First suppose that (i) (Y, X) is an edge in the graph. Then Figure 4.2 and Tables 4.1 and 4.2 give an example of $\mathcal{G}$ with $\text{X}(\mathcal{G}) \not\subseteq \text{Y}(\mathcal{G})$, so that $\text{Y}(\mathcal{G}) \neq \text{X}(\mathcal{G})$. For example, the edge from $\text{MPO}(\mathcal{G})$ to $\text{PO}(\mathcal{G})$ is labelled with MODS $\mathcal{G}_{15}$, which has $\mathcal{S} = \{s_1, s_5\}$ (as defined in Table 4.1) and $\text{MPO}(\mathcal{G}_{15}) = \{\alpha, \beta\}$, which is a strict subset of $\text{PO}(\mathcal{G}_{15}) = \{\alpha, \beta, \delta\}$: see Table 4.2.

Now suppose that (ii) (Y, X) is in $\mathcal{L}$, and let (Y, Z) be an edge on a path from Y to X in the graph. We have, by the previous part of the theorem, that for all $\mathcal{G}$, $\text{Y}(\mathcal{G}) \subseteq \text{Z}(\mathcal{G}) \subseteq \text{X}(\mathcal{G})$. By Case (i), there exists $\mathcal{G}$ such that $\text{Y}(\mathcal{G}) \neq \text{Z}(\mathcal{G})$, and so $\text{Y}(\mathcal{G}) \neq \text{X}(\mathcal{G})$.

We now consider the general case. The construction of $\mathcal{C}$ implies that there exists $\text{W} \in \mathcal{C}$ with, for all $\mathcal{G}$, $\text{X}(\mathcal{G}) \cap \text{Y}(\mathcal{G}) = \text{W}(\mathcal{G})$. Also, (W, X) is in $\mathcal{L}$. By Case (ii) there exists $\mathcal{G}$ with $\text{W}(\mathcal{G}) \neq \text{X}(\mathcal{G})$, i.e., $\text{X}(\mathcal{G}) \cap \text{Y}(\mathcal{G}) \neq \text{X}(\mathcal{G})$, so that $\text{X}(\mathcal{G}) \not\subseteq \text{Y}(\mathcal{G})$, as required.

Examples $\mathcal{G}_{13}$ and $\mathcal{G}_{14}$ (see Table 4.2) show that $\text{NSO}(\mathcal{G})$, $\text{NOPSO}(\mathcal{G})$, $\text{NO}(\mathcal{G})$ and

PSO($\mathcal{G}$) can be empty. In fact, it is easy to construct a single $\mathcal{G}$ such that they are all empty; for example, if there are three scenarios, with sets of optimal decisions $\{\alpha, \beta\}$, $\{\alpha, \gamma\}$ and $\{\beta, \gamma\}$. ▪

Table 4.2: Classes in different examples – we consider six different MODS $\mathcal{G}$ over set of decisions $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$, using different sets of scenarios from Table 4.1. For example, $\mathcal{G}_{12}$ involves set of scenarios $\{s_1, s_2\}$ with orderings as defined in Table 4.1.

	NSO($\mathcal{G}$)	NOPSO($\mathcal{G}$)	NO($\mathcal{G}$)	PSO($\mathcal{G}$)	MPO-EXT($\mathcal{G}$)	EXT($\mathcal{G}$)
$\mathcal{G}12$	α, β	α, β	α, β	α, β	α, β	α, β
$\mathcal{G}13$	$\emptyset$	$\emptyset$	$\emptyset$	δ	β, δ	β, δ
$\mathcal{G}14$	$\emptyset$	$\emptyset$	α	$\emptyset$	α	α
$\mathcal{G}15$	$\emptyset$	α, β	α, β	α, β	α, β	α, β
$\mathcal{G}345$	$\emptyset$	$\emptyset$	$\emptyset$	δ	α, δ	α, γ, δ
$\mathcal{G}567$	$\emptyset$	$\emptyset$	$\emptyset$	γ	β, γ, δ	β, γ, δ

	MPO′($\mathcal{G}$)	MPO($\mathcal{G}$)	PO′($\mathcal{G}$)	PO($\mathcal{G}$)	CSD($\mathcal{G}$)	CD($\mathcal{G}$)
$\mathcal{G}12$	α, β	α, β	α, β	α, β	α, β	α, β
$\mathcal{G}13$	β, δ	α, β, δ	β, δ	α, β, δ	β, γ, δ	$\mathcal{A}$
$\mathcal{G}14$	α	α	α	α, β, γ	α	α, β, γ
$\mathcal{G}15$	α, β	α, β	α, β	α, β, δ	α, β	α, β, δ
$\mathcal{G}345$	α, δ	α, δ	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$
$\mathcal{G}567$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$

4.4 Subclass Simplifications Under Extra Conditions

In this section, we consider three important extra conditions, which are:

(i) if there exists a decision that is optimal in all scenarios;

(ii) if for any decision there exists a most favourable scenario; and

(iii) if the orders are all total orders.

We then show how the subclass relationships between the optimality classes simplify under each condition.

4.4.1 When there exists a Necessarily Optimal Decision

The following result, Proposition 4.8, considers the case when there exists a necessarily optimal decision. Part (i) of the result below implies that the graph of subset relations collapses to a chain, with $\mathrm{NO}(\mathcal{G})$ equalling $\mathrm{CSD}(\mathcal{G})$, and $\mathrm{PO}(\mathcal{G}) = \mathrm{CD}(\mathcal{G})$. Part (ii) implies that either $\mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G}) = \mathrm{PSO}(\mathcal{G})$ or $\mathrm{NOPSO}(\mathcal{G}) = \emptyset$. Part (iii) implies that when there exists a necessarily strictly optimal decision, the set of optimal decisions are the same in each scenario and all the classes including $\mathrm{NSO}(\mathcal{G})$, $\mathrm{NO}(\mathcal{G})$, $\mathrm{PSO}(\mathcal{G})$, $\mathrm{PO}(\mathcal{G})$, $\mathrm{CSD}(\mathcal{G})$, $\mathrm{CD}(\mathcal{G})$ and any O_s and SO_s collapse into one class, i.e., are all equal.

Proposition 4.8 » Necessarily optimal decision exists

Suppose, for some MODS $\mathcal{G} = \langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\}\rangle$, *that* $\mathrm{NO}(\mathcal{G}) \neq \emptyset$, *and let* s *be an arbitrary scenario in* $\mathcal{S}$. *Then,*

(i) $\mathrm{NSO}(\mathcal{G}) \subseteq \mathrm{SO}_s \subseteq \mathrm{PSO}(\mathcal{G}) \subseteq (\mathrm{NO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G}) = \mathrm{CSD}(\mathcal{G})) \subseteq \mathrm{O}_s \subseteq (\mathrm{PO}(\mathcal{G}) = \mathrm{CD}(\mathcal{G}))$.

(ii) For $\mathrm{X} \in \{\mathrm{NSO}(\mathcal{G}), \mathrm{NOPSO}(\mathcal{G}), \mathrm{PSO}(\mathcal{G}), \mathrm{SO}_s\}$, *either* $\mathrm{X} = \emptyset$ *or* $\mathrm{X} = \mathrm{NO}(\mathcal{G})$.

(iii) If $\mathrm{NSO}(\mathcal{G}) \neq \emptyset$ *then all the classes* $\mathrm{NSO}(\mathcal{G})$, SO_s, $\mathrm{PSO}(\mathcal{G})$, $\mathrm{NO}(\mathcal{G})$, $\mathrm{CSD}(\mathcal{G})$ $\mathrm{MPO}(\mathcal{G})$, $\mathrm{EXT}(\mathcal{G})$, O_s, $\mathrm{PO}(\mathcal{G})$ *and* $\mathrm{CD}(\mathcal{G})$ *are equal.* ◇

Proof:

(i) First we show $\mathrm{NO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G})$. Since $\mathrm{NO}(\mathcal{G})$ is non-empty, and any element of $\mathrm{NO}(\mathcal{G})$ is optimal in all scenarios then we have $\alpha \in \mathrm{MPO}(\mathcal{G})$ if and only if $\alpha \in \mathrm{MPO}(\mathcal{G})$.

Now we show $\mathrm{NO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G})$. Suppose $\alpha \in \mathrm{EXT}(\mathcal{G})$. Then there exists some sufficient sequence $s_1, \ldots, s_k$ such that $A_{s_1,\ldots,s_k} \ni \alpha$, with all elements of $A_{s_1,\ldots,s_k}$ being necessarily equivalent. Since $\mathrm{NO}(\mathcal{G})$ is non empty, then let β be an element of $\mathrm{NO}(\mathcal{G})$. Since $\mathrm{EXT}(\mathcal{G})$ is closed under improvement, then $\beta \in A_{s_1,\ldots,s_k}$, which means that α and β are necessarily equivalent. Therefore we have that $\alpha \in \mathrm{NO}(\mathcal{G})$, and therefore $\mathrm{EXT}(\mathcal{G}) \subseteq \mathrm{NO}(\mathcal{G})$. Since in the general case we also have $\mathrm{NO}(\mathcal{G}) \subseteq \mathrm{EXT}(\mathcal{G})$, then we have $\mathrm{NO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G})$.

Here we show $\mathrm{NO}(\mathcal{G}) = \mathrm{CSD}(\mathcal{G})$. Suppose that $\alpha \in \mathrm{CSD}$, and since $\mathrm{NO}(\mathcal{G})$ is non-empty, let β be some element in $\mathrm{NO}(\mathcal{G})$. We have that $\beta \preccurlyeq_{\mathrm{N}} \alpha$. But

since α is undominated w.r.t $\prec_N$, we don't have that $\beta \prec_N \alpha$, so we must have that $\alpha \equiv_N \beta$, and hence $\alpha \in \text{NO}(\mathcal{G})$. Therefore we have that $\text{CSD}(\mathcal{G}) \subseteq \text{NO}(\mathcal{G})$ and since in the general case we also have $\text{NO}(\mathcal{G}) \subseteq \text{CSD}(\mathcal{G})$, then we have $\text{NO}(\mathcal{G}) = \text{CSD}(\mathcal{G})$.

Now we show $\text{PO}(\mathcal{G}) = \text{CD}(\mathcal{G})$. Suppose that $\alpha \in \text{CD}(\mathcal{G})$, and since $\text{NO}(\mathcal{G})$ is non-empty, let β be some element in $\text{NO}(\mathcal{G})$. By definition of $\text{CD}(\mathcal{G})$, there exists some scenario s such that $\alpha \preccurlyeq_s \beta$. Since β is optimal in every scenario, then α is in O_s, and therefore α is in $\text{PO}(\mathcal{G})$. Therefore we have that $\text{CD}(\mathcal{G}) \subseteq \text{PO}(\mathcal{G})$, and since in the general case we also have $\text{PO}(\mathcal{G}) \subseteq \text{CD}(\mathcal{G})$, then we have $\text{PO}(\mathcal{G}) = \text{CD}(\mathcal{G})$.

The rest of the result follows from Proposition 4.3, parts (iv) and (v), i.e., $\text{NO}(\mathcal{G}) \subseteq \text{O}_s \subseteq \text{PO}(\mathcal{G})$ and $\text{NSO}(\mathcal{G}) \subseteq \text{SO}_s \subseteq \text{PSO}(\mathcal{G})$.

(ii) Let X be any of $\{\text{NSO}(\mathcal{G}), \text{NOPSO}(\mathcal{G}), \text{PSO}(\mathcal{G}), \text{SO}_s\}$, and suppose that X is non-empty. By (i), $\text{X} \subseteq \text{NO}(\mathcal{G})$, and so there exists some α in X and $\text{NO}(\mathcal{G})$. If β is any element of $\text{NO}(\mathcal{G})$, then $\beta \equiv_N \alpha$, which implies that $\beta \in \text{X}$, since X is closed under improvement. Since β was arbitrary, this shows that $\text{NO}(\mathcal{G}) \subseteq \text{X}$, and hence $\text{NO}(\mathcal{G}) = \text{X}$.

(iii) Suppose $\text{NSO}(\mathcal{G}) \neq \emptyset$. Now suppose we have some decision β in $\text{CD}(\mathcal{G})$, but β not in $\text{NSO}(\mathcal{G})$. Since $\text{NSO}(\mathcal{G})$ is not empty, then we have some $\alpha \in \text{NSO}(\mathcal{G})$, i.e., for all $\beta' \in \mathcal{A} \setminus [\alpha]_N$, $\alpha \prec_{NS} \beta'$. Therefore, for all $\beta' \in \mathcal{A} \setminus [\alpha]_N$, for all $s' \in \mathcal{S}$, $\alpha \prec_{s'} \beta'$. However this contradicts $\beta \in \text{CD}(\mathcal{G})$, since there is no scenario $s \in \mathcal{S}$ such that $\beta \preccurlyeq_s \alpha$. Therefore we have that, if $\text{NSO}(\mathcal{G}) \neq \emptyset$, then $\text{CD}(\mathcal{G}) \subseteq \text{NSO}(\mathcal{G})$ and therefore $\text{NSO}(\mathcal{G}) = \text{CD}(\mathcal{G})$, which gives us that the classes all collapse into one. ∎

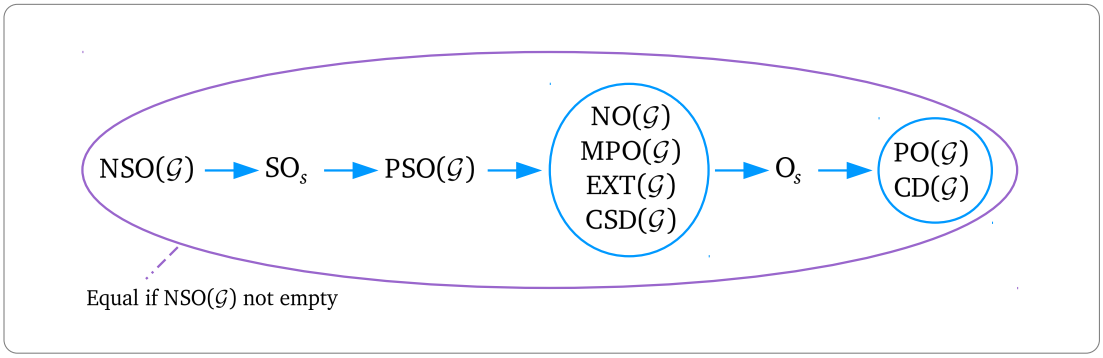


Figure 4.3: Proposition 4.8 gives us that when $\text{NO}(\mathcal{G})$ is non-empty, the graph of subset relations simplifies as shown, where circled classes show classes that collapse to equality.

4.4.2 Most Favourable Scenario for a Decision

We now look at how the subclass hierarchy simplifies when there is a *most favourable scenario* for every decision $\alpha \in \mathcal{A}$, defined as follows:

Definition 4.21 » Most favourable scenario for α
We say that scenario s' is a *most favourable scenario for* α, if

— for all $\beta \in \mathcal{A}$, for all $s \in \mathcal{S}$, $[\alpha \preccurlyeq_s \beta \Rightarrow \alpha \preccurlyeq_{s'} \beta]$ and $[\alpha \prec_s \beta \Rightarrow \alpha \prec_{s'} \beta]$ «

Let us look at an example.

Example 4.5 ► Most favourable scenario example**.**
Consider MODS $\mathcal{G} = \mathcal{G}_{15}$*, with set of decisions* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$*, set of scenarios* $\mathcal{S} = \{s_1, s_5\}$*, where the associated total pre-orders, originally given in Table 4.1, are recalled as follows.*

$\preccurlyeq_{s_1}$	$\preccurlyeq_{s_5}$
$\alpha\ \beta$	$\alpha\ \beta\ \delta$
γ	γ
δ	

We can see that

- *Scenario* s_1 *is a most favourable scenario for* α*,* β *and* γ
- *Scenario* s_2 *is a most favourable scenario for* δ*.* ▲

Another example of a most favourable scenario for a decision occurs in interval-valued soft constraints problems, as we will see in Section 4.6.2. The following result gives us the subclass relationships between the classes when there exists most favourable scenarios for all decisions (see also Figure 4.4).

Proposition 4.9 » Most favourable scenario exists

Suppose, for some MODS $\mathcal{G}$*, that for every decision* $\alpha \in \mathcal{A}$ *there exists a most favourable scenario.*

Then $\mathrm{CSD}(\mathcal{G}) = \mathrm{PSO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G})$ *and* $\mathrm{CD}(\mathcal{G}) = \mathrm{PO}(\mathcal{G})$.

Hence we then have the following relationships:

$\mathrm{NSO}(\mathcal{G}) \subseteq \mathrm{NO}(\mathcal{G}) \subseteq (\mathrm{PSO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G}) = \mathrm{PO}'(\mathcal{G}) = \mathrm{CSD}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G})) \subseteq (\mathrm{PO}(\mathcal{G}) = \mathrm{CD}(\mathcal{G}))$. ◇

Proof: First we show that $\mathrm{CSD}(\mathcal{G}) \subseteq \mathrm{PSO}(\mathcal{G})$. Suppose $\alpha \in \mathrm{CSD}(\mathcal{G})$. Let s be a most favourable scenario for α. Consider any other decision $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$. Since $\alpha \in \mathrm{CSD}(\mathcal{G})$ there exists some scenario s' with $\alpha \prec_{s'} \beta$. Hence, $\alpha \prec_s \beta$. This shows that α is strictly optimal in scenario s, and thus is in $\mathrm{PSO}(\mathcal{G})$. Therefore we have $\mathrm{CSD}(\mathcal{G}) \subseteq \mathrm{PSO}(\mathcal{G})$, and so $\mathrm{CSD}(\mathcal{G}) = \mathrm{PSO}(\mathcal{G})$.

Similarly, now we show $\mathrm{CD}(\mathcal{G}) \subseteq \mathrm{PO}(\mathcal{G})$. Suppose $\alpha \in \mathrm{CD}(\mathcal{G})$. Let s be a most favourable scenario for α. Consider any other decision $\beta \in \mathcal{A}$. Since $\alpha \in \mathrm{CD}(\mathcal{G})$ there exists some scenario s' with $\alpha \preccurlyeq_{s'} \beta$. Hence, $\alpha \preccurlyeq_s \beta$. This shows that α is optimal in scenario s, and thus is in $\mathrm{PO}(\mathcal{G})$. Therefore we have $\mathrm{CD}(\mathcal{G}) \subseteq \mathrm{PO}(\mathcal{G})$, and so $\mathrm{PO}(\mathcal{G}) = \mathrm{CD}(\mathcal{G})$.

$\mathrm{CSD}(\mathcal{G}) \subseteq \mathrm{PSO}(\mathcal{G})$ implies also $\mathrm{CSD}(\mathcal{G}) \subseteq \mathrm{MPO}(\mathcal{G})$, and therefore $\mathrm{CSD}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G})$.

Now we show that $\mathrm{MPO}(\mathcal{G}) \subseteq \mathrm{CSD}(\mathcal{G})$. Suppose there exists some α which is in $\mathrm{MPO}(\mathcal{G})$ but not in $\mathrm{CSD}(\mathcal{G})$. $\alpha \notin \mathrm{CSD}(\mathcal{G})$ implies that there exists β such that $\beta \prec_{\mathrm{N}} \alpha$. This implies that $\mathrm{Opt}(\beta) \supseteq \mathrm{Opt}(\alpha)$, and so $\mathrm{Opt}(\beta) = \mathrm{Opt}(\alpha)$, since $\alpha \in \mathrm{MPO}(\mathcal{G})$. Also, $\beta \prec_{\mathrm{N}} \alpha$ implies that there exists some scenario s with $\beta \prec_s \alpha$. Let s' be a most favourable scenario for β. Then $\beta \prec_{s'} \alpha$, and also, since $\beta \in \mathrm{PO}(\mathcal{G})$ it can be seen that β is optimal in s'. However, $\beta \prec_{s'} \alpha$ implies that α is not optimal in s', which contradicts that $\mathrm{Opt}(\beta) = \mathrm{Opt}(\alpha)$ (i.e., that β and α are optimal in the same scenarios). Thus the earlier assumption that $\mathrm{MPO}(\mathcal{G}) \not\subseteq \mathrm{CSD}(\mathcal{G})$ is incorrect, so we have $\mathrm{MPO}(\mathcal{G}) \subseteq \mathrm{CSD}(\mathcal{G})$, and therefore $\mathrm{CSD}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G})$. ∎

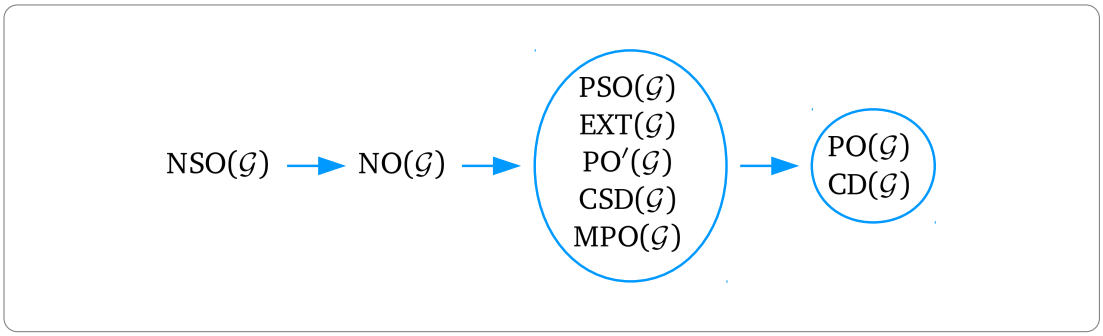


Figure 4.4: Proposition 4.9 gives us that when there exists a most favourable scenario for every decision, the graph of subset relations simplifies as shown, where circled classes show classes that collapse to equality.

4.4.3 When Scenarios Totally Order Decisions

In this section, we consider the case when the ordering in each scenario is a total order, i.e., for all s, for all α, β, either $\alpha \preccurlyeq_s \beta$ or $\beta \preccurlyeq_s \alpha$ (see Definition 2.2). The class hierarchy simplifies as follows (see Figure 4.5).

Proposition 4.10 » Scenarios as total orders

Suppose, for some MODS $\mathcal{G}$*, that for every scenario* $s \in \mathcal{S}$*,* $\preccurlyeq_s$ *is a total order. Then we have:*

(i) $\mathrm{NSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G})$*,* $\mathrm{PSO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G})$*,* $\mathrm{CSD}(\mathcal{G}) = \mathrm{CD}(\mathcal{G})$*, and for every scenario* s*,* $\mathrm{SO}_s = \mathrm{O}_s$*.*

We have the following relationships between the classes:

$(\mathrm{NSO}(\mathcal{G}) = \mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G})) \subseteq (\mathrm{PSO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G})) \subseteq (\mathrm{CSD}(\mathcal{G}) = \mathrm{CD}(\mathcal{G}))$.

(ii) If there exists a necessarily optimal decision α*, then all the classes* $\mathrm{NSO}(\mathcal{G})$*,* $\mathrm{NOPSO}(\mathcal{G})$*,* $\mathrm{NO}(\mathcal{G})$*,* $\mathrm{PSO}(\mathcal{G})$*,* $\mathrm{PO}'(\mathcal{G})$*,* $\mathrm{PO}(\mathcal{G})$*,* $\mathrm{CSD}(\mathcal{G})$ *and* $\mathrm{CD}(\mathcal{G})$ *are equal to* $\{\alpha\}$*.*

(iii) If, for every decision $\alpha \in \mathcal{A}$*, there exists a most favourable scenario for* α *then we have:*

$(\mathrm{NSO}(\mathcal{G}) = \mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G})) \subseteq (\mathrm{PSO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G}) = \mathrm{CSD}(\mathcal{G}) = \mathrm{CD}(\mathcal{G}))$.

◇

Proof:

(i) Recall that $\alpha \in \mathrm{NSO}(\mathcal{G})$ if and only if for all $s \in \mathcal{S}$, $\alpha \prec_s \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{N}}$, and recall that $\alpha \in \mathrm{NO}(\mathcal{G})$ if and only if for all $s \in \mathcal{S}$, $\alpha \preccurlyeq_s \beta$ for all $\beta \in \mathcal{A}$. Since for all $s \in \mathcal{S}$, $\preccurlyeq_s$ is a total order, then we have that $\equiv_s$ is equality. Also, since $\equiv_{\mathrm{N}} = \bigcap_{s \in \mathcal{S}} \equiv_s$, then $\equiv_{\mathrm{N}}$ is equality, therefore for all $\alpha, \beta \in \mathcal{A}$, if $\alpha \preccurlyeq_{\mathrm{N}} \beta$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$, then $\alpha = \beta$. This means that definitions for $\mathrm{NSO}(\mathcal{G})$ and $\mathrm{NO}(\mathcal{G})$ are now the same, since for each $\alpha \in \mathcal{A}$, $[\alpha]_{\mathrm{N}}$ consists of just $\{\alpha\}$. Similarly, the definitions coincide for $\mathrm{PSO}(\mathcal{G})$ and $\mathrm{PO}(\mathcal{G})$, and $\mathrm{CSD}(\mathcal{G})$ and $\mathrm{CD}(\mathcal{G})$, and SO_s and O_s, which gives us that $(\mathrm{NSO}(\mathcal{G}) = \mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G})) \subseteq (\mathrm{PSO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G})) \subseteq (\mathrm{CSD}(\mathcal{G}) = \mathrm{CD}(\mathcal{G}))$.

(ii) Suppose there exists some $\alpha \in \mathcal{A}$ such that α is in $\mathrm{NO}(\mathcal{G})$. Then, from (i), and from Proposition 4.8 (i), where $\mathrm{NO}(\mathcal{G}) = \mathrm{CSD}(\mathcal{G})$, we have that all the classes $\mathrm{NSO}(\mathcal{G})$, $\mathrm{NOPSO}(\mathcal{G})$, $\mathrm{NO}(\mathcal{G})$, $\mathrm{PSO}(\mathcal{G})$, $\mathrm{PO}'(\mathcal{G})$, $\mathrm{PO}(\mathcal{G})$, $\mathrm{CSD}(\mathcal{G})$ and $\mathrm{CD}(\mathcal{G})$ are equal.

Now suppose that there is some $\beta \in \mathcal{A}$ such that β is in $\mathrm{NO}(\mathcal{G})$. Then we have that $\alpha \preccurlyeq_{\mathrm{N}} \beta$ and $\beta \preccurlyeq_{\mathrm{N}} \alpha$, i.e., $\alpha \equiv_{\mathrm{N}} \beta$. Since each $\preccurlyeq_s$ is a total order, therefore $\alpha \equiv_{\mathrm{N}} \beta$ if and only if $\alpha = \beta$. Therefore we have that $\mathrm{NO}(\mathcal{G}) = \{\alpha\}$.

(iii) This follows from Proposition 4.9, since if there exists a most favourable scenario for each decision, then we have $\mathrm{CSD}(\mathcal{G}) = \mathrm{PSO}(\mathcal{G})$, which gives us that $\mathrm{PSO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) = \mathrm{MPO}(\mathcal{G}) = \mathrm{EXT}(\mathcal{G}) = \mathrm{CSD}(\mathcal{G}) = \mathrm{CD}(\mathcal{G})$. ∎

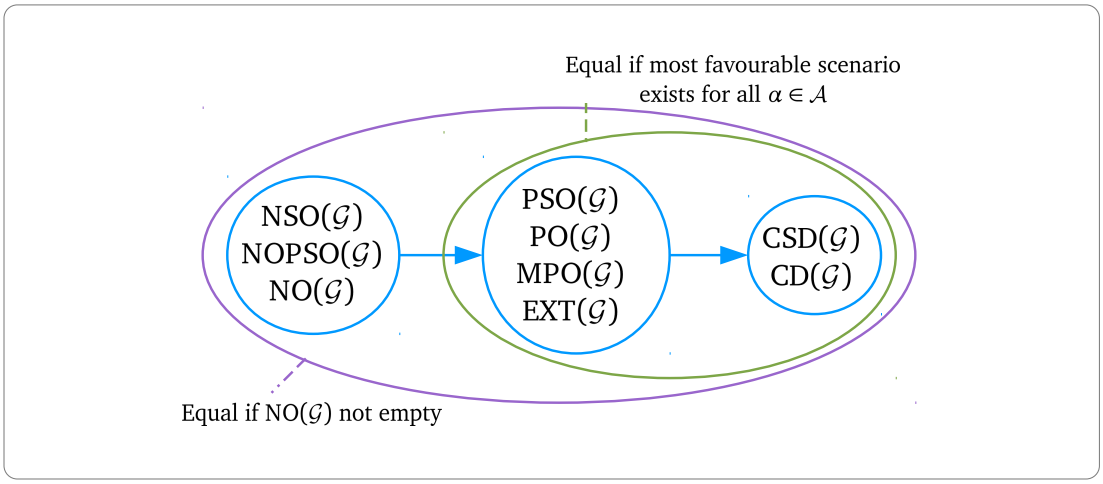


Figure 4.5: Proposition 4.10 gives us that when for each scenario the ordering is a total order, the graph of subset relations simplifies as shown, where circled classes show classes that collapse to equality.

4.5 A Multi-criteria Example

As mentioned in Section 4.2, the set of scenarios in the MODS framework can also have other interpretations. In a multi-criteria decision making or multi-criteria optimisation problem [KR76], the decisions or alternatives in the problem are evaluated over a number of different criteria, and the task is usually to choose a decision or set of decisions that optimise these evaluations. In this section, we look at a small example of the MODS framework in multi-criteria decision making, where the scenarios in the MODS framework represent criteria.

We assume that the objective in each criterion is to minimise. Let MODS $\mathcal{M} = \langle \mathcal{A}, \mathcal{C}, \{\preccurlyeq_i : i \in \mathcal{C}\}\rangle$, where $\mathcal{A}$ is a non-empty, finite set of decisions, $\mathcal{C}$ is a non-empty finite set of criteria, and $\{\preccurlyeq_i : i \in \mathcal{C}\}$ is the set of orderings in each criteria. For $\alpha, \beta \in \mathcal{A}$, we have that $\alpha \preccurlyeq_{\mathrm{N}} \beta$ if and only if $\alpha \preccurlyeq_i \beta$, for all $i \in \mathcal{C}$, i.e., α is at least as good as β in every criterion. We also have that $\alpha \prec_{\mathrm{N}} \beta$ if and only if $\alpha \preccurlyeq_{\mathrm{N}} \beta$ and $\beta \not\preccurlyeq_{\mathrm{N}} \alpha$, i.e., α is at least as good as β in every criterion, and strictly better in some. We can see here that the necessarily dominates relation and the necessarily strictly dominates relation correspond the Weak Pareto and Pareto dominance relations respectively, as given in Section 2.4. Also, since CSD($\mathcal{M}$) are the decisions that are undominated with respect to $\prec_{\mathrm{N}}$, which in this instance of the framework is the Pareto dominance relation, then these are the decisions that correspond to what is known in multi-criteria decision making literature as *Pareto optimal* or *non-dominated* (see Definition 2.18).

Let us look at an example of a multi-criteria MODS and the resulting optimality classes.

Example 4.6 ► Multi-criteria MODS example**.**
Consider the following MODS $\mathcal{M}$*, where the set of decisions is* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$ *and the set of criteria is* $\mathcal{C} = \{1, 2, 3\}$.

For decision $\alpha \in \mathcal{A}$*, let* $v(\alpha) = (\alpha_1, \alpha_2, \alpha_3)$ *be the preference vector for* α*, consisting of the evaluation of* α *over each criterion* $i \in \mathcal{C}$.

Suppose the preference vectors for each decision are as follows:

- $v(\alpha) = (4, 1, 1)$
- $v(\beta) = (3, 2, 1)$
- $v(\gamma) = (2, 1, 2)$

- $v(\delta) = (1, 3, 1)$

These preference vectors give us an ordering of the decisions for each criterion as follows:

$\preccurlyeq_1$	$\preccurlyeq_2$	$\preccurlyeq_3$
δ	$\alpha\ \gamma$	$\alpha\ \beta\ \delta$
γ	β	γ
β	δ	
α		

Since, for each decision there exists a scenario in which it is optimal, and given that $\mathrm{PO}(\mathcal{M}) \subseteq \mathrm{CD}(\mathcal{M})$*, we have:*

- $\mathrm{PO}(\mathcal{M}) = \{\alpha, \beta, \gamma, \delta\} = \mathcal{A}$
- $\mathrm{CD}(\mathcal{M}) = \mathcal{A}$

Also, any of the decisions can strictly dominate any other decision in some scenario, and since $\mathrm{PO}'(\mathcal{M}) = \mathrm{PO}(\mathcal{M}) \cap \mathrm{CSD}(\mathcal{M})$*, so we have:*

- $\mathrm{CSD}(\mathcal{M}) = \{\alpha, \beta, \gamma, \delta\} = \mathcal{A}$
- $\mathrm{PO}'(\mathcal{M}) = \mathcal{A}$

Both α *and* δ *are optimal in a maximal subset of criteria, i.e.,* α *is optimal in each of the criteria in* $\{2, 3\}$ *and* δ *is optimal in each of the criteria in* $\{1, 3\}$*, so we have:*

- $\mathrm{MPO}(\mathcal{M}) = \{\alpha, \delta\}$
- $\mathrm{MPO}'(\mathcal{M}) = \{\alpha, \delta\}$

β *is not in* $\mathrm{EXT}(\mathcal{M})$*, since there is no sequence of criteria for which the resulting equivalence class considered by* $\mathrm{EXT}(\mathcal{M})$ *contains* β*. We have such sequences of criteria for* α, γ *and* δ*, for example,* $(3, 2)$*,* $(2, 1)$ *and* $(3, 1)$ *respectively, therefore we have:*

- $\mathrm{EXT}(\mathcal{M}) = \{\alpha, \gamma, \delta\}$
- $\text{MPO-EXT}(\mathcal{M}) = \{\alpha, \delta\}$

We can see that δ *strictly dominates all other decisions in criterion* 1*. Since no other decision can strictly dominate all other decisions in some criterion, we have:*

- $\mathrm{PSO}(\mathcal{M}) = \{\delta\}$

Finally, we can see that there are no decisions that necessarily dominate all others decisions in every criterion, so we have:

► $\mathrm{NSO}(\mathcal{M}) = \mathrm{NOPSO}(\mathcal{M}) = \mathrm{NO}(\mathcal{M}) = \emptyset$ ▲

4.6 Discussion

In this section, we look at one possible way to use the general MODS framework and the subclass relationships between the optimality classes to order the decisions. We also conclude the chapter with some further discussion on the optimality classes.

4.6.1 Using Subclass Relationships to Order Decisions

Considering the task where we wish to show a list of decisions to a decision maker, how should we order this list so that the more interesting decisions are first? Again, we consider that we only have qualitative information as expressed in a given MODS $\mathcal{G} = \langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\}\rangle$, and that we only want to use this purely qualitative information, (e.g., we don't want to convert the orderings to numerical rankings, and we don't want to reason based on numbers of the scenarios). We now outline one approach for presenting such a list of decisions to a decision maker.

Firstly, for MODS $\mathcal{G}$, for all $\alpha \in \mathcal{A}$, let $\mathrm{X}_{\mathcal{G}}(\alpha)$ be the minimal optimality class that contains α . Based on this, we define a new preference relation as follows:

Definition 4.22 » Minimal-class dominates
For all $\alpha, \beta \in \mathcal{A}$, we say that decision α *minimal-class dominates* β, written as $\alpha \prec_{\mathcal{L}} \beta$, if and only if

— $\mathrm{X}_{\mathcal{G}}(\alpha)$ is a strict subset of $\mathrm{X}_{\mathcal{G}}(\beta)$. «

This relation $\prec_{\mathcal{L}}$ is compatible with relation $\prec_N$ and we can therefore show the user decisions in an order compatible with the union of these two relations, i.e., $\prec_{\mathcal{L}} \cup \prec_N$. Let us look at an example.

Example 4.7 ► Minimal classes example**.**
Consider MODS $\mathcal{G} = \mathcal{G}_{123}$*, with set of decisions* $\mathcal{A} = \{\alpha, \beta, \gamma, \delta\}$*, set of scenarios* $\mathcal{S} = \{s_1, s_2, s_3\}$*, and with associated total pre-orders given in Table 4.1, which we recall as follows:*

$\preccurlyeq_{s_1}$	$\preccurlyeq_{s_2}$	$\preccurlyeq_{s_3}$
$\alpha\ \beta$	$\alpha\ \beta$	δ
γ	δ	γ
δ	γ	β
		α

Considering the optimality classes already calculated in Example 4.2, along with the optimality classes defined consequently, gives us the following.

- $\mathrm{NSO}(\mathcal{G}) = \mathrm{NOPSO}(\mathcal{G}) = \mathrm{NO}(\mathcal{G}) = \emptyset$
- $\mathrm{PSO}(\mathcal{G}) = \{\delta\}$
- $\mathrm{MPO\text{-}EXT}(\mathcal{G}) = \mathrm{MPO}'(\mathcal{G}) = \mathrm{EXT}(\mathcal{G}) = \mathrm{PO}'(\mathcal{G}) = \{\beta, \delta\}$
- $\mathrm{MPO}(\mathcal{G}) = \mathrm{PO}(\mathcal{G}) = \{\alpha, \beta, \delta\}$
- $\mathrm{CSD}(\mathcal{G}) = \{\beta, \gamma, \delta\}$
- $\mathrm{CD}(\mathcal{G}) = \{\alpha, \beta, \gamma, \delta\}$

From these optimality classes, we determine the minimal class for each decision. Figure 4.6 shows the minimal class for the decisions on the subclass diagram.

- $\mathrm{X}_{\mathcal{G}}(\alpha) = \mathrm{MPO}(\mathcal{G})$
- $\mathrm{X}_{\mathcal{G}}(\beta) = \mathrm{MPO\text{-}EXT}(\mathcal{G})$
- $\mathrm{X}_{\mathcal{G}}(\gamma) = \mathrm{CSD}(\mathcal{G})$
- $\mathrm{X}_{\mathcal{G}}(\delta) = \mathrm{PSO}(\mathcal{G})$

From the minimal classes, we can determine the minimal-class dominance relation $\prec_{\mathcal{L}}$:

- $\prec_{\mathcal{L}} = \{(\delta, \beta), (\delta, \alpha), (\delta, \gamma), (\beta, \alpha), (\beta, \gamma)\}$

Also from the scenario orderings, we can see that the relation $\prec_{\mathrm{N}}$ is given by

- $\prec_{\mathrm{N}} = \{(\beta, \alpha)\}$

Using both these relations, we can show the decisions to a decision maker in a manner that is compatible with these relations along with a description of the optimality class to which it belongs, e.g.,

1. δ *is possibly strictly optimal*
2. β *is maximally possibly optimal and is an extreme decision*

3a. α *is maximally possibly optimal*

3b. γ *can strictly dominate every other decision in some scenario*

In the case where we have two decisions that do not dominate each other according to $\prec_{\mathcal{L}} \cup \prec_{\mathrm{N}}$*, e.g., in this example* α *and* γ*, and since we could present with each decision the description of the minimal optimality class to which it belongs; this would allow the decision maker to choose between these decisions considering the optimality class, which might depend on e.g., the context of the problem or their own attitudes to risk. For example, a decision maker who does not mind some risk may prefer decision* α *over decision* γ*, even though* α *is not in* $\mathrm{CSD}(\mathcal{G})$*, (and decision* γ *is), decision* α *is more likely to be optimal for some scenarios, since it is in* $\mathrm{MPO}(\mathcal{G})$ *(and decisions* γ *is not).* ▲

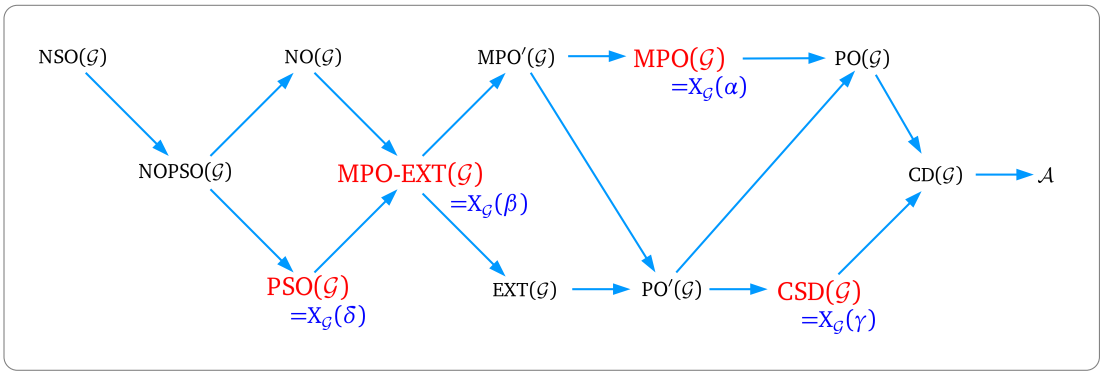


Figure 4.6: For Example 4.7, for MODS $\mathcal{G} = \mathcal{G}_{123}$, we show the minimal classes for α, β, γ and δ on the subclass relationship diagram.

4.6.2 Relationship with Interval Valued Soft Constraints

In this section, we discuss the connection between the MODS framework and the resulting notions of optimality with the optimality classes for interval-valued soft constraints [GPR+10b]. In a soft constraints problem (see Section 3.3), a soft constraint is a function which associates a preference value to scoped tuples of domain values (or assignments). In an interval valued soft constraints problem, instead of associating a single preference value with an assignment, an interval valued soft constraint associates an interval of preference values with an assignment. First we define an interval-valued soft constraint problem.

Definition 4.23 » Interval-valued soft constraint problem (IVSCSP)
An interval-valued soft constraint problem is a semiring constraints network $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$ with totally ordered c-semiring $\langle E, +, \times, 0, 1 \rangle$ (see Definition 3.3), where $\mathcal{C}_S$ is a set of interval-valued soft constraints, and where an interval-valued soft constraint i_V, with scope $V \subseteq \mathcal{X}$, is a function $i_V : \mathcal{D}(V) \to E \times E$. «

For an interval-valued soft constraint i_V, for some tuple t, we let $l_{i_V}(t)$ denote the first component of $i_V(t)$, and we let $u_{i_V}(t)$ denote the second component, i.e., we have $i_V(t) = (l_{i_V}(t), u_{i_V}(t))$.

The preference level for an assignment is given as follows.

Definition 4.24 » Assignment preference level (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, the *preference level* $\rho(t)$ of an assignment $t \in \mathcal{D}(\mathcal{X})$ is given by

— $\rho(t) = [L(t), U(t)]$

where $L(t) = \times_{i_V \in \mathcal{C}_S} l_{i_V}(t)$, and $U(t) = \times_{i_V \in \mathcal{C}_S} u_{i_V}(t)$ «

That is, for any tuple t, the preference interval of t is given by a lower bound and an upper bound, where the lower bound is the combination of all the lower bounds of the preference intervals associated to t by the interval-valued soft constraints, and the upper bound of t is the combination of all the upper bounds of the preference intervals associated to t by the constraints.

The intervals in each constraint represent some sort imprecision or uncertainty around the actual preference value. A *scenario* in an interval-valued soft constraint problem occurs as a result of choosing a preference value from the interval.

Definition 4.25 » Scenario (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, a *scenario* P' of P is a problem $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'_S, S \rangle$, such that for all $i_V \in \mathcal{C}_S$, let s_V be a soft constraint in $\mathcal{C}'_S$ such that $s_V(t) \in [l_{i_V}(t), u_{i_V}(t)]$. «

Then, given a particular scenario, the preference level of an assignment in that scenario is defined follows.

Definition 4.26 » Preference level in a scenario (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, the preference level $\rho(t)$ of a solution t in a given scenario $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'_S, S \rangle$ is given by

— $\rho_{P'}(t) = \bigtimes_{s_V \in \mathcal{C}'_S} s_V(t)$ «

We can say that a solution dominates or strictly dominates another in a given scenario as follows.

Definition 4.27 » Dominance (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, for a scenario $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'_S, S \rangle$, a solution $t \in \mathcal{D}(\mathcal{X})$ *dominates* a solution $t' \in \mathcal{D}(\mathcal{X})$ in scenario P' if and only if

— $\rho_{P'}(t) \succeq_S \rho_{P'}(t')$ «

Definition 4.28 » Strict dominance (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, for a scenario $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'_S, S \rangle$, a solution $t \in \mathcal{D}(\mathcal{X})$ *strictly dominates* a solution $t' \in \mathcal{D}(\mathcal{X})$ in scenario P' if and only if

— $\rho_{P'}(t) \succ_S \rho_{P'}(t')$ «

Given a scenario, an optimal solution in that scenario is defined as follows.

Definition 4.29 » Optimal solution in a scenario (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, for a scenario $P' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}'_S, S \rangle$, a solution $t \in \mathcal{D}(\mathcal{X})$ is optimal in scenario P' if and only if

— for all $t' \in \mathcal{D}(\mathcal{X})$, $\rho_{P'}(t) \succeq_S \rho_{P'}(t')$ «

We have different notions of optimality for interval-valued soft constraints problems (Definitions 6 and 7 from [GPR+10b]).

Definition 4.30 » Necessarily optimal (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, an assignment is *necessarily optimal* if and only if it is optimal in all scenarios. «

The set of necessarily optimal elements is denoted by $NO(P)$.

Definition 4.31 » Possibly optimal (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, an assignment is *possibly optimal* if and only if it is optimal in some scenario. «

The set of possibly optimal elements is denoted by $PO(P)$.

We also have the notions of necessarily dominates and necessarily strictly dominates in interval-valued soft constraints (Definitions 15 and 16 from [GPR$^+$10b]).

Definition 4.32 » Necessarily dominates (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, an assignment $t \in \mathcal{D}(\mathcal{X})$ necessarily dominates an assignment t' in $\mathcal{D}(\mathcal{X})$ if and only if

— for all scenarios P', $\rho_{P'}(t) \succeq_S \rho_{P'}(t')$ «

Definition 4.33 » Necessarily strictly dominates (IVSCSP)
For an interval-valued soft constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_S, S \rangle$, an assignment $t \in \mathcal{D}(\mathcal{X})$ necessarily strictly dominates an assignment t' in $\mathcal{D}(\mathcal{X})$ if and only if

— for all scenarios P', $\rho_{P'}(t) \succ_S \rho_{P'}(t')$ «

The undominated solutions with respect to necessarily dominates are denoted by $NDTOP(P)$ and the undominated solutions with respect to necessarily strictly dominates are denoted $NSDTOP(P)$. The results in Theorem 12 in [GPR$^+$10b] gives us different subclass relationships between the optimality classes for the interval valued soft constraints. For example, the first part of Theorem 12 gives us that $NO(P) \subseteq NDTOP(P) \subseteq NSDTOP(P)$, and the second part gives us that $PO(P) \subseteq NSDTOP(P)$.

We can see that the scenarios of an interval-valued soft constraint problem correspond to the scenarios in the MODS framework, and that some of the different notions of optimality for interval-valued soft constraints and the MODS framework coincide. The $NO(P)$ class for interval-valued soft constraints corresponds to the class NO($\mathcal{G}$) in the MODS framework, similarly for the $PO(P)$ and PO($\mathcal{G}$) classes. Since the notions of necessarily dominates and necessarily strictly dominates for interval-valued soft constraints correspond to $\preccurlyeq_{\text{N}}$ and $\prec_{\text{NS}}$ respectively in the MODS framework, and we have that $NSDTOP(P)$ corresponds to CD($\mathcal{G}$), and $NDTOP(P)$ corresponds to CSD($\mathcal{G}$).

In Theorem 8 in [GPR$^+$10a], we have the definition of a scenario Q_s, which is the scenario where every preference associated to a tuple of s is set to its lower bound and the preferences of all other tuples are set to their upper bound. In our MODS framework, this corresponds to the definition of a most favourable scenario for a decision, as given in Section 4.4.2. Proposition 4.9 gives us in the MODS framework that NO($\mathcal{G}$) $\subseteq$ (PSO($\mathcal{G}$) = CSD($\mathcal{G}$)) $\subseteq$ (PO($\mathcal{G}$) = CSD($\mathcal{G}$)), then we have for strictly monotonic interval-valued soft constraints problems (where the c-semiring is strictly monotonic) that $NO(P) \subseteq NDTOP \subseteq (PO(P) = NSDTOP(P))$, which corresponds to the results in the first four parts in Theorem 12 in [GPR$^+$10b]. In addition, if we have that NO($\mathcal{G}$) is non-empty, then we have from Proposition 4.8 in the MODS framework that PSO($\mathcal{G}$) $\subseteq$ (NO($\mathcal{G}$) = CSD($\mathcal{G}$)) $\subseteq$ (PO($\mathcal{G}$) = CD($\mathcal{G}$)), which gives us that PSO($\mathcal{G}$) = NO($\mathcal{G}$) = CSD($\mathcal{G}$) = PO($\mathcal{G}$) = CD($\mathcal{G}$), so for the interval valued soft constraints problem we have that $NO(P) = NDTOP(P) = PO(P) = NSDTOP(P)$. Therefore we can see that some of the notions of optimality from [GPR$^+$10b] also apply in the more general context of the MODS framework.

4.7 Chapter Conclusion

In this chapter, we looked at some notions for optimality for decision making with qualitative information, in situations where we want to present a set of decisions to a decision maker that are in some way "optimal", or "preferable" to the other decisions. If there exists a necessarily optimal decision, then being necessarily optimal seems the most natural notion of optimality; most of the optimality classes we consider collapse to NO($\mathcal{G}$) in this case: see Proposition 4.8. However, more generally, there is no unique most appropriate notion of optimality in all contexts. If one of the scenarios represents the true preference ordering, then PO($\mathcal{G}$) gives the decisions that could be optimal. However, we might eliminate decisions not in CSD($\mathcal{G}$) (each of which is inferior to one that is in CSD($\mathcal{G}$)), leading to PO$'$($\mathcal{G}$). Where this class is large, we may refine this to consider more special decisions in different ways, and based on different intuitions, leading to MPO$'$($\mathcal{G}$), EXT($\mathcal{G}$) and MPO-EXT($\mathcal{G}$), the latter being the most specific class we consider that is always non-empty.

The notions of optimality in the MODS framework are partly inspired by the notions of optimality for interval-valued constraints [GPR$^+$10b] as seen in Section 4.6.2, however we assume only qualitative or ordinal information. The MODS framework has a connection to *decision-making under uncertainty* (DMU) [Sav54], where the decision-making task involves the consideration of multiple possible states that

can occur and the outcomes associated to the decisions in these states. [HK95] classifies various different types of uncertainty in decision problems, where we have precise, ambiguous or no knowledge on the probabilities or outcomes of different problems. Often such DMU frameworks assume that there is precise knowledge, for example, a probability distribution that assigns probabilities to the states, such as in expected utility framework [vNM47]. However our framework is more closely related to decision-making under *complete uncertainty*, or *ignorance* [AH72, Mas79, LKM10, CM10] where there is no information assumed as to which states are more likely than others.

We only considered qualitative orderings, and purely qualitative notions of optimality, such as in [FP99]. If in each scenario there is instead a numerical utility or cost function over decisions (or if one converts the qualitative orderings to numerical rankings) then other natural notions of optimality are available, e.g., for positive preferences we could use Maxmin [Raw71], Leximin [BJ88], or Minimax regret [Sav51], and for negative preferences we could use Lexicographic Min-Max [Ehr96] or take the min-sum of costs, such as is done in Weighted CSP [RvBW06]; in fact, any of the preference relations discussed in Section 2.3 that can utilise quantitative preferences could be used. Furthermore, if we have, for example, a probability distribution over scenarios, then we can consider the decisions that have highest probability of being optimal [Pea88], or those that maximise expected utility [vNM47].

Chapter 5

Sorted-Pareto Dominance and Soft Constraints

5.1 Introduction

The notion of Pareto optimality originated in social welfare and economic theory [Par97], and the Pareto dominance relation is widely used in that field and many other related decision making fields. In a general decision-making context, a decision Pareto dominates another if it is strictly preferred in at least one aspect of the decision and at least as good in all other aspects (see Section 2.4) where an aspect could be: a voter in collective decision making, a state of the world in decision making under uncertainty [CM10], or a criterion in multi-criteria decision-making [KR76]. However, a problem with Pareto dominance is its lack of discriminatory power, as many comparisons between pairs of decisions do not result in dominance, which in turn leads to a large number of non-dominated or Pareto optimal solutions. Therefore, it is desirable to look at relations that extend Pareto dominance, where the extending relation has more power when comparing decisions, thus leading to a smaller set of non-dominated decisions that are all still Pareto optimal.

In this chapter we look at an extension to the Pareto dominance relation, called Sorted-Pareto dominance. As shown in Section 2.6, this relation is extended by the Lexicographic-Max relation [Ehr05] and Leximin [DFP96b, PSS06], which compares two decisions by lexicographically comparing the worst evaluations of the decisions. However these Lexicographic relations places excessive emphasis on the worst evaluations (as it ignores the better evaluations when comparing two decisions, except if the decisions have the same worst evaluation) whereas Sorted-Pareto compares decisions considering all evaluations. Also, Sorted-Pareto assumes only a qualitative or ordinal scale, and therefore does not rely on quantitative information to compare decisions (as in, for example, the quantitative preference relations discussed in Section 2.6).

The remainder of the chapter is outlined as follows. In Section 5.2 we revisit the definitions for the Sorted-Pareto dominance relation, giving some properties that characterise the relation, and in Section 5.3 we give a semantics for Sorted-Pareto dominance in terms of a preference relation that is compatible with any order-preserving mapping that maps a qualitative scale to a quantitative one. In Section 5.4, we examine Sorted-Pareto dominance in the context of a general constraints problem framework, and in Section 5.5 we describe some backtracking search and depth first branch and bound algorithms for searching for a set of Sorted-Pareto optimal solutions. Section 5.6 details our implementation of a constraints solver for Sorted-Pareto framework and the algorithms in our Soft Constraints solver, and

in Section 5.7 we discuss some results obtained when solving general constraint problem instances using this implementation, where we examine the solver execution times for the algorithms developed as well as looking at the resulting sizes of the Sorted-Pareto optimal sets for different types of problems. In Section 5.8 we describe some extensions to the Sorted-Pareto dominance relation and look at some further experimental results. Section 5.9 highlights some similar work in this area, and Section 5.10 concludes with some discussion.

5.2 Sorted-Pareto Dominance

We assume a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$ from Section 2.3, and we recall the definition as follows.

Recall » General Decision Making Problem (Definition 2.11)
A *multi-aspect decision problem* is a tuple $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where:

- $\mathcal{A}$ is a finite set of decisions, alternatives or choices,
- $\mathcal{S} = \{1, \ldots, m\}$ is a finite set of decision *aspects*, where each $i \in \mathcal{S}$ labels some preferential aspect of the problem, and for which p_i is a function that specifies the preference value of each decision, i.e., $p_i : \mathcal{A} \to T$,
- T is a scale of preference values, where $\leq$ is a total order on T. «

We also recall for each decision $\alpha \in \mathcal{A}$ we have a preference vector $v(\alpha) = (\alpha_1, \ldots, \alpha_m)$, and $v(\alpha)^{\uparrow} = (\alpha_{(1)}, \ldots, \alpha_{(m)})$ is such that $\alpha_{(1)} \leq \cdots \leq \alpha_{(m)}$. We recall the following notation for preference vector dominance from Section 2.2.

Recall » Preference vector dominance (Definition 2.14)
For any two preference vectors $v(\alpha)$ and $v(\beta)$, of size m, we have that

(i) $v(\alpha) \leq v(\beta)$ if and only if $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$,

(ii) $v(\alpha) < v(\beta)$ if and only if $\alpha_i \leq \beta_i$ for all $i \in \{1, \ldots, m\}$, and there exists $j \in \{i, \ldots, m\}$ such that $\alpha_j < \beta_j$.

(iii) $v(\alpha) = v(\beta)$ if and only if $\alpha_i = \beta_i$ for all $i \in \{1, \ldots, m\}$. «

The Sorted-Pareto dominance relation, which we introduced in Section 2.5 is a preference relation which can be used in the ordering of decisions with multiple

qualitative preference levels. We assume decision making situations where the decision aspects use the same scale, for example in the situation where the aspects correspond to different voters or experts, or different criteria with commensurate scales. However, even in situations where evaluations of aspects may not be on the same scale, then there are methods for modifying or normalising the different scales, e.g., such as the joint ordinal scale [LM95, MMO02], so that the scales are commensurate and of equal importance. We also assume that each of the aspects are of equal importance, or in other terms, that the ordering of the preference vector is irrelevant. This naturally occurs in decision making situations such as, group decision making where all experts are considered equal, or in decision making under uncertainty where there is no information on which state will occur.

Now we recall the definitions of the Sorted-Pareto dominance relations from Section 2.5

Recall » Weak Sorted-Pareto dominance (Definition 2.19)
For all $\alpha, \beta \in \mathcal{A}$, α *Weak Sorted-Pareto dominates* β, written as $\alpha \preccurlyeq_{\text{SP}} \beta$, iff.

— $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$ «

Recall » Sorted-Pareto dominance (Definition 2.20)
For all $\alpha, \beta \in \mathcal{A}$, α *Sorted-Pareto dominates* β, written as $\alpha \prec_{\text{SP}} \beta$, if and only if

— $v(\alpha)^{\uparrow} < v(\beta)^{\uparrow}$ «

Recall » Sorted-Pareto equivalence (Definition 2.21)
For all $\alpha, \beta \in \mathcal{A}$, α is *Sorted-Pareto equivalent* to β, written as $\alpha \equiv_{\text{SP}} \beta$, if and only if

— $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$ «

5.2.1 Properties of Sorted-Pareto Dominance

In this section, we give some general properties of the Sorted-Pareto dominance relations. First, we have the following result.

Proposition 5.1 » Properties of Sorted-Pareto Dominance

For a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, we have for the Sorted-Pareto relations $\preccurlyeq_{\text{SP}}$, $\prec_{\text{SP}}$ and $\equiv_{\text{SP}}$

(i) Relation $\preccurlyeq_{\text{SP}}$ is a preorder on $\mathcal{A}$, i.e., it is reflexive and transitive.

(ii) Relation $\prec_{\text{SP}}$ is a strict preorder on $\mathcal{A}$, i.e., it is irreflexive and transitive.

(iii) Relation $\equiv_{\text{SP}}$ is an equivalence relation on $\mathcal{A}$, i.e., it is reflexive, transitive, and symmetric. ◇

Proof:

(i) First we show that $\preccurlyeq_{\text{SP}}$ is transitive. Suppose $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \preccurlyeq_{\text{SP}} \gamma$. Then by definition of $\preccurlyeq_{\text{SP}}$, $\upsilon(\alpha)^{\uparrow} \leq \upsilon(\beta)^{\uparrow}$ and $\upsilon(\beta)^{\uparrow} \leq \upsilon(\gamma)^{\uparrow}$. By transitivity of $\leq$, $\upsilon(\alpha)^{\uparrow} \leq \upsilon(\gamma)^{\uparrow}$, which by definition of $\preccurlyeq_{\text{SP}}$, implies that $\alpha \preccurlyeq_{\text{SP}} \gamma$. Therefore $\preccurlyeq_{\text{SP}}$ is transitive.

Now we show that $\preccurlyeq_{\text{SP}}$ is reflexive. For any $\alpha \in \mathcal{A}$, consider its sorted preference vector, $\upsilon(\alpha)^{\uparrow} = (\alpha_{(1)}, \ldots, \alpha_{(m)})$. Since T is a totally ordered scale, then for any scale value $\upsilon \in T$, $\upsilon \leq \upsilon$, which means for any $\alpha_{(i)} \in \upsilon(\alpha)^{\uparrow}$, $\alpha_{(i)} \leq \alpha_{(i)}$, i.e., for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \alpha_{(i)}$. Therefore, by definition of the vector dominance relation $\leq$, we have $\upsilon(\alpha)^{\uparrow} \leq \upsilon(\alpha)^{\uparrow}$, which by definition of $\preccurlyeq_{\text{SP}}$ implies that $\alpha \preccurlyeq_{\text{SP}} \alpha$. Therefore $\preccurlyeq_{\text{SP}}$ is reflexive.

(ii) This follows from Proposition 2.2 and the properties stated in Section 2.2.1, however we include a more detailed proof here to aid in the intuition of the relation.

First we show that $\prec_{\text{SP}}$ is transitive. Suppose $\alpha \prec_{\text{SP}} \beta$ and $\beta \prec_{\text{SP}} \gamma$. Then by definition of $\preccurlyeq_{\text{SP}}$, we have that $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \preccurlyeq_{\text{SP}} \gamma$, which by transitivity of $\preccurlyeq_{\text{SP}}$, means that $\alpha \preccurlyeq_{\text{SP}} \gamma$. From this, we must have either $\alpha \equiv_{\text{SP}} \gamma$ or $\alpha \prec_{\text{SP}} \gamma$. If we have that $\alpha \equiv_{\text{SP}} \gamma$, then for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} = \gamma_{(i)}$, and since we have $\beta \preccurlyeq_{\text{SP}} \gamma$, then for all $i \in \{1, \ldots, m\}$, $\beta_{(i)} \leq \gamma_{(i)}$ and therefore for all $i \in \{1, \ldots, m\}$, $\beta_{(i)} \leq \alpha_{(i)}$ This contradicts $\alpha \prec_{\text{SP}} \beta$, since there exists no $i \in \{1, \ldots, m\}$ such that $\alpha_{(i)} < \beta_{(i)}$. Therefore we must have that $\alpha \prec_{\text{SP}} \gamma$, and thus $\prec_{\text{SP}}$ is transitive.

Since, $\alpha \prec_{\text{SP}} \alpha$ if and only if $\alpha \preccurlyeq_{\text{SP}} \alpha$ and $\alpha \not\preccurlyeq_{\text{SP}} \alpha$ is a contradiction, we have that $\prec_{\text{SP}}$ is irreflexive.

(iii) This follows from Proposition 2.3 and the properties stated in Section 2.2.1, however as above we include a more detailed proof here to aid in the intuition of the relation.

First we show that $\equiv_{\text{SP}}$ is reflexive. For any $\alpha \in \mathcal{A}$, consider its sorted preference vector, $v(\alpha)^{\uparrow} = (\alpha_{(1)}, \ldots, \alpha_{(m)})$. Since T is a totally ordered scale, then for any scale value $v \in T$, $v = v$, which means for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} = \alpha_{(i)}$. Therefore, by definition of vector equality, we have $v(\alpha)^{\uparrow} = v(\alpha)^{\uparrow}$, which by definition of $\equiv_{\text{SP}}$ implies that $\alpha \equiv_{\text{SP}} \alpha$. Therefore $\equiv_{\text{SP}}$ is reflexive.

Now we show that $\equiv_{\text{SP}}$ is symmetric. Suppose we have $\alpha \equiv_{\text{SP}} \beta$. Then we have that $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$. By symmetry of equality, we have $v(\beta)^{\uparrow} = v(\alpha)^{\uparrow}$, which means that we have $\beta \equiv_{\text{SP}} \alpha$.

Now we show that $\equiv_{\text{SP}}$ is transitive. Suppose we have that $\alpha \equiv_{\text{SP}} \beta$ and $\beta \equiv_{\text{SP}} \gamma$. Then we have that $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$ and $v(\beta)^{\uparrow} = v(\gamma)^{\uparrow}$. By transitivity of equality, we have $v(\alpha)^{\uparrow} = v(\gamma)^{\uparrow}$, which means that we have $\alpha \equiv_{\text{SP}} \gamma$. ▪

Now we give an alternative characterisation of Weak Sorted-Pareto dominance, which connects the relation to the orderings defined in Definition 2.1 of [BS06] and Definition 11 of [PS05], which looks at the problem of finding minimal paths on ordered graphs.

First we give some notation. Let σ be a permutation function on $\{1, \ldots, m\}$, i.e., it is a *bijective* function, $\sigma : \{1, \ldots, m\} \to \{1, \ldots, m\}$. Relaxing notation, we extend this definition of a permutation function to also use over preference vectors; a permutation function σ is a bijective function on a preference vector $v(\alpha)$ that gives a reordering of the preference vector, i.e., let $\sigma\big(v(\alpha)\big) = (\alpha_{\sigma(1)}, \alpha_{\sigma(2)}, \ldots, \alpha_{\sigma(m)})$.

We have the following result.

Proposition 5.2 » Sorted-Pareto permutation result

For all $\alpha, \beta \in \mathcal{A}$, $\alpha \preccurlyeq_{\text{SP}} \beta$ if and only if there exists a vector permutation function σ such that, $\sigma\big(v(\alpha)\big) \leq v(\beta)$. ◇

Proof: Suppose that $\alpha \preccurlyeq_{\text{SP}} \beta$. Then we have that $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$, i.e., for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$. Let σ' be a permutation function such that $\sigma'\big(v(\beta)^{\uparrow}\big) = v(\beta)$, i.e., we have that $\sigma'\big(v(\beta)^{\uparrow}\big) = \big(\beta_{\sigma'((1))}, \beta_{\sigma'((2))}, \ldots, \beta_{\sigma'((m))}\big) = (\beta_1, \beta_2, \ldots \beta_m)$. Let

$A = \sigma'\big(v(\alpha)^{\uparrow}\big)$. We have $A_i \leq \beta_{\sigma'((1))}$, for all $i \in \{1,\ldots,m\}$, and thus $A_i \leq \beta_i$ for all $i \in \{1,\ldots,m\}$. This gives us that $A \leq v(\beta)$ and since A is a permuted vector of $v(\alpha)^{\uparrow}$, which is in turn a permuted vector of $v(\alpha)$, then we have that there exists a permutation σ such that $\sigma\big(v(\alpha)\big) \leq v(\beta)$.

Now we prove the converse. Suppose that $\sigma\big(v(\alpha)\big) \leq v(\beta)$. This implies $\big(\sigma\big(v(\alpha)\big)\big)^{\uparrow} \leq \big(v(\beta)\big)^{\uparrow}$ using the proof of Proposition 2.4, which is equivalent to $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$, i.e., $\alpha \preccurlyeq_{\text{SP}} \beta$.

∎

5.2.2 A Characterisation of Sorted-Pareto as a Relation on Multisets

In the preceding discussions, we viewed the Sorted-Pareto preference relation as an ordering over a set of decisions $\mathcal{A}$, where each decision $\alpha \in \mathcal{A}$ has a preference vector $v(\alpha)$ of m values from a given scale T. In Section 2.5.1, we also looked at the induced Sorted-Pareto dominance relation over the preference vector space T^m.

In this section, we now show how the Sorted-Pareto ordering can be characterised when viewed as an ordering on M^T, the set of *multisets* of T. To facilitate this characterisation, we define the following.

Definition 5.1 » Unaffected by permutations
Let us say that an ordering $\preccurlyeq$ on T^m is *unaffected by permutations* if, for any two vectors $\mathbf{u}$, $\mathbf{v} \in T^m$, and for any two permutation functions σ and σ',

- $\mathbf{u} \preccurlyeq \mathbf{v} \iff \sigma(\mathbf{u}) \preccurlyeq \sigma'(\mathbf{v})$. «

In this case, $\preccurlyeq$ can be represented as an ordering on M^T, the set of multisets of T, rather than vectors of T. Orderings that are unaffected by permutations are important, for instance, in the context of soft constraints, where the ordering of the soft constraints is taken to be irrelevant, so we can view the input as being a multiset of soft constraints. Its definition immediately implies that the Sorted-Pareto dominance relation is unaffected by permutations. In fact, it can be easily seen that it is the setwise smallest such relation that extends the Pareto dominance relation.

Proposition 5.3 » Sorted-Pareto and unaffected by permutations

If an ordering $\preccurlyeq$ on T^m extends Pareto dominance and is unaffected by permutations, then

— $\preccurlyeq \supseteq \preccurlyeq_{\text{SP}}$ ◇

Proof: Suppose for $\mathbf{u}, \mathbf{v} \in T^m$, we have $\mathbf{u} \preccurlyeq_{\text{SP}} \mathbf{v}$. By definition of $\preccurlyeq_{\text{SP}}$, we have that $\mathbf{u}^{\uparrow} \leq \mathbf{v}^{\uparrow}$. Therefore, since $\preccurlyeq$ extends Pareto dominance, we have that $\mathbf{u}^{\uparrow} \preccurlyeq \mathbf{v}^{\uparrow}$. Since $\mathbf{u}^{\uparrow}$ is a permutation of $\mathbf{u}$ and $\mathbf{v}^{\uparrow}$ is a permutation of $\mathbf{v}$, and since $\preccurlyeq$ is unaffected by permutations, we have $\mathbf{u} \preccurlyeq \mathbf{v}$. ∎

In the remainder of this section, we consider the induced Sorted-Pareto ordering on M^T. First, for multiset $A \in M^T$, let $A^{\uparrow}$ be the vector in T^m which has the values of A in non-descending order, i.e., it is the sorted vector of the values of multiset A. We induce the Weak Sorted-Pareto dominance relation on M^T as follows.

Definition 5.2 » Sorted-Pareto dominance on M^T
For multisets $A, B \in M^T$,

- $\emptyset \preccurlyeq_{\text{SP}} \emptyset$
- $A \preccurlyeq_{\text{SP}} B \Leftrightarrow A^{\uparrow} \leq B^{\uparrow}$ «

From this definition, we have that the Sorted-Pareto relation $\preccurlyeq_{\text{SP}}$ is a *partial order* on M^T (see Definition 2.2).

Proposition 5.4 » Sorted-Pareto dominance on M^T

$\preccurlyeq_{\text{SP}}$ is a partial order on M^T, i.e., it is reflexive, transitive and antisymmetric. ◇

Proof: Reflexivity and transitivity of $\preccurlyeq_{\text{SP}}$ follow from reflexivity and transitivity of $\leq$. For antisymmetry, we have $A \preccurlyeq_{\text{SP}} B \preccurlyeq_{\text{SP}} A$ implies $A^{\uparrow} \leq B^{\uparrow} \leq A^{\uparrow}$, and hence $A^{\uparrow} = B^{\uparrow}$, which implies $A = B$. ∎

Consider the following two properties for orderings of M^T . The first property relates the ordering on T with the ordering on singleton multisets. The second is a kind of independence ($\uplus$ is multiset sum).

Property 5.1. For all $x, y \in T$, $x \leq y \Rightarrow \{x\} \preccurlyeq \{y\}$

Property 5.2. For all $A, B, C \in M^T$, $A \preccurlyeq B \Rightarrow A \uplus C \preccurlyeq B \uplus C$

Proposition 5.5 » Sorted-Pareto relation on T-multisets

The Sorted-Pareto ordering $\preccurlyeq_{\text{SP}}$ on multisets of T is the unique minimal preorder on M^T satisfying Properties 5.1 and 5.2.

That is, $\preccurlyeq_{\text{SP}}$ satisfies Properties 5.1 and 5.2, and if $\preccurlyeq$ is an ordering on M^T satisfying Properties 5.1 and 5.2 then for $A, B \in M^T$, $A \preccurlyeq_{\text{SP}} B \Rightarrow A \preccurlyeq B$. ◇

Proof: First, $\preccurlyeq_{\text{SP}}$ on multisets of T satisfies Property 5.1 by definition of $\preccurlyeq_{\text{SP}}$.

Now we show that $\preccurlyeq_{\text{SP}}$ satisfies Property 5.2. First, let us derive a consequence of Property 5.2.

Property (*) If for all $j = 1, \ldots, k$ we have $A^j \preccurlyeq B^j$, then $A \preccurlyeq B$, where A is the multiset union of the A^j's and similarly for B.

To prove (*), we first prove the case when $k = 2$, i.e., $A^1 \preccurlyeq B^1$ and $A^2 \preccurlyeq B^2$ implies $A^1 \uplus A^2 \preccurlyeq B^1 \uplus B^2$. (*) follows from this by an inductive/iterative argument. To prove the property for $k = 2$ we just apply Property 5.2 twice, using transitivity: Assume $A^1 \preccurlyeq B^1$ and $A^2 \preccurlyeq B^2$ Then Property 5.2 implies $A^1 \uplus A^2 \preccurlyeq B^1 \uplus A^2$. Also Property 5.2 implies $B^1 \uplus A^2 \preccurlyeq B^1 \uplus B^2$, Then by transitivity of $\preccurlyeq$, $A^1 \uplus A^2 \preccurlyeq B^1 \uplus B^2$.

To complete the proof: Suppose that $A \preccurlyeq_S B$, and that $\preccurlyeq$ satisfies properties 5.1 and 5.2 (and hence (*)). We have for all i, $A_{(i)} \leq B_{(i)}$. Thus by Property 5.1, $\{A_{(i)}\} \preccurlyeq \{B_{(i)}\}$.

Property (*) then implies $A \preccurlyeq B$, as required. ∎

Another characterisation of Sorted-Pareto is given by Property 5.3 below.

Property 5.3. For all $A, B \in M^T$, $A \preccurlyeq B$ if and only if

(I) $\min(A) \leq \min(B)$ and

(II) $A - \{\min(A)\} \preccurlyeq B - \{\min(B)\}$.

The following result in Proposition 5.6 shows that the Sorted-Pareto ordering is the unique ordering, that only compares multisets of equal cardinality, satisfying

Property 5.3.

Proposition 5.6 » Sorted-Pareto dominance on equal cardinality multisets

Let $\preccurlyeq$ be an ordering on M^T that only compares multisets of equal cardinality, i.e., such that $A \preccurlyeq B \Rightarrow |A| = |B|$.

Then $\preccurlyeq$ satisfies Property 5.3 if and only if $\preccurlyeq = \preccurlyeq_{\text{SP}}$. ◇

Proof: Let $P(\preccurlyeq)$ be the condition of satisfying Property 5.3, i.e., for all $A, B \in M^T$, $A \preccurlyeq B$ if and only if

(I) $\min(A) \leq \min(B)$; and

(II) $A - \{\min(A)\} \preccurlyeq B - \{\min(B)\}$.

We need to show that $P(\preccurlyeq)$ holds iff $\preccurlyeq = \preccurlyeq_{\text{SP}}$.

Let $A_{(1)} = \min(A)$, $A_{(2)}$ is the second smallest element of A, and so on. Let $A[k]$ be A with the k smallest elements removed. Let's assume that $P(\preccurlyeq)$ holds. We have $A \preccurlyeq B$ iff $A_{(1)} \leq B_{(1)}$ and $A[1] \preccurlyeq B[1]$. We also have $A[1] \preccurlyeq B[1]$ iff $A_{(2)} \leq B_{(2)}$ and $A[2] \preccurlyeq B[2]$, and so on.

Thus $A \preccurlyeq B$ iff $A_{(1)} \leq B_{(1)}$ and $A[1] \preccurlyeq B[1]$ iff $A_{(1)} \leq B_{(1)}$ and $A_{(2)} \leq B_{(2)}$ and $A[2] \preccurlyeq B[2]$ iff $A_{(1)} \leq B_{(1)}$ and $A_{(2)} \leq B_{(2)}$ and $A_{(3)} \leq B_{(3)}$ and $A[3] \preccurlyeq B[3]$ iff (continuing) for all $i, A_{(i)} \leq B_{(i)}$, i.e., iff $A \preccurlyeq_{\text{SP}} B$.

We've shown that $P(\preccurlyeq)$ holds only if $\preccurlyeq = \preccurlyeq_{\text{SP}}$. To complete the proof we just need to show that $P(\preccurlyeq_{\text{SP}})$ holds.

$P(\preccurlyeq_{\text{SP}})$ states that for all multisets $A, B \in M^T$ we have $A \preccurlyeq_{\text{SP}} B$ iff $A_{(1)} \leq B_{(1)}$ and $A[1] \preccurlyeq_{\text{SP}} B[1]$. Both $\Rightarrow$ and $\Leftarrow$ follow fairly easily from the definition of Sorted Pareto. ∎

5.3 A Semantics for Sorted-Pareto Dominance

We now look at a semantics for the Sorted-Pareto dominance. We assume a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where each decision $\alpha \in \mathcal{A}$ is characterised by its preference vector $v(\alpha)$. One way of comparing decisions using these preference values is to map the qualitative scale values onto quantitative values, for example,

representing some sort of numerical cost. To do this, one can define a weights function f on the scale values, e.g., $f : T \to \mathbb{R}^+$, where the function is *monotonic* with respect to the ordering of the scale. We recall the definition of monotonic function from Section 2.6 here.

Recall » Monotonic function (Definition 2.32)
For some ordered scale T, ordered by $\leq$, a function $f : T \to \mathbb{R}^+$ is *monotonic* with respect to scale T, if and only if

— $u \leq v \Rightarrow f(u) \leq f(v)$, for all $u, v \in T$ «

Therefore, for our set of decisions $\mathcal{A}$, and using some weights function $f : T \to \mathbb{R}^+$, the decisions can be ordered by summing and comparing the preference vectors associated with each decision. This is the Min-sum of weights ordering, as seen in Section 2.3, and we recall the definition here.

Recall » Min-sum preferred (Definition 2.34)
For all $\alpha, \beta \in \mathcal{A}$, for some function $f : T \to \mathbb{R}^+$, α is *Min-sum preferred* (with respect to f) to β, written as $\alpha \leq_f \beta$, if and only if

— $\sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$ «

This order relation $\leq_f$ is a total preorder on the set of decisions $\mathcal{A}$, but for different mappings (i.e., different $f : T \to \mathbb{R}^+$), the resulting orders can be different. Let us look at an example.

Example 5.1 ► Sum of weights example**.**
Consider some multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, where we have:

- $\mathcal{A} = \{\alpha, \beta, \gamma, \delta, \varepsilon, \zeta\}$*, i.e., there are 6 decisions,*
- $\mathcal{S} = \{1, 2\}$*, i.e., there are two aspects for which we have preferences values,*
- $T = \{low, med, hi\}$*, which is ordered by $\leq$, and the scale represents some negative preferences which we wish to minimise.*

Suppose that the resulting preference vectors are:

- $v(\alpha) = (low, low)$*;*
- $v(\beta) = (low, med)$*;*
- $v(\gamma) = (low, hi)$*;*

- $v(\delta) = (med, med)$;
- $v(\varepsilon) = (med, hi)$; and
- $v(\zeta) = (hi, hi)$.

Let f_1 and f_2 be two weight functions, e.g., cost functions, that are applied to the preference vectors associated with the decisions in the set $\mathcal{A} = \{\alpha, \beta, \gamma, \delta, \varepsilon, \zeta\}$.

f_1 and f_2 are defined as follows, where the ordered pair $\big(i, f(i)\big)$ denotes the mapping from $i \in T$ to $f(i) \in \mathbb{R}^+$:

- $f_1 = \big\{(hi, 6), (med, 3), (low, 2)\big\}$
- $f_2 = \big\{(hi, 5), (med, 4), (low, 1)\big\}$

The following table shows the resulting sum of weights, for both f_1 and f_2 when applied to the decisions in $\mathcal{A}$.

	$\sum_{i=1}^{m} f_1(v_i)$	$\sum_{i=1}^{m} f_2(v_i)$
$v(\alpha) = (low, low)$	4	2
$v(\beta) = (low, med)$	5	5
$v(\gamma) = (low, hi)$	8	6
$v(\delta) = (med, med)$	6	8
$v(\varepsilon) = (med, hi)$	9	9
$v(\zeta) = (hi, hi)$	12	10

Using the sum of weights for f_1 and f_2, the resulting orderings $\leq_{f_1}$ and $\leq_{f_2}$ over $\mathcal{A}$ are:

- $\leq_{f_1}$: $\alpha \leq_{f_1} \beta \leq_{f_1} \delta \leq_{f_1} \gamma \leq_{f_1} \varepsilon \leq_{f_1} \zeta$
- $\leq_{f_2}$: $\alpha \leq_{f_2} \beta \leq_{f_2} \gamma \leq_{f_2} \delta \leq_{f_2} \varepsilon \leq_{f_2} \zeta$

Both f_1 and f_2 are monotonic with respect to scale T, i.e., for all $v, v' \in T$, we have $u \leq v \Rightarrow f_1(u) \leq f_1(v)$ and $u \leq v \Rightarrow f_2(u) \leq f_2(v)$, but the resulting orderings $\leq_{f_1}$ and $\leq_{f_2}$ over $\mathcal{A}$ are different.

We can see in ordering $\leq_{f_1}$ that

- $\delta \leq_{f_1} \gamma$,

but we can see in ordering $\leq_{f_2}$ that

- $\gamma \leq_{f_2} \delta$ *and* $\delta \not\leq_{f_2} \gamma$ ▲

When it is possible to provide a weights function (like f_1 or f_2) to map the scale values to some quantitative measure, then it is easy to compare decisions by using the sum of these weights. However, sometimes it is not possible to create this quantitative mapping, e.g., when this information is not available or is uncertain, or it may be hard to elicit such values from a decision maker [Lar92, MMO02], so we consider a different order that does not rely on this quantitative information.

First, we define the following set of functions.

Definition 5.3 » Set F_T of monotonic functions
For some ordered scale T, let F_T be the set of all weight functions $f : T \to \mathbb{R}^+$ such that $f \in F_T$ if and only if f is monotonic with respect to T, i.e., $u \leq v \Rightarrow f(u) \leq f(v)$ for all $u, v \in T$. «

We now define an order relation $\leq_F$ on $\mathcal{A}$ as follows:

Definition 5.4 » Relation $\leq_F$
For all $\alpha, \beta \in \mathcal{A}$,

$$\alpha \leq_F \beta \iff [\alpha \leq_f \beta, \text{ for all } f \in F_T] \quad \text{«}$$

This relation is the intersection of all possible order relations $\leq_f$, for all monotonic functions f defined on T, i.e., we have that

$$\leq_F = \bigcap_{f \in F_T} \leq_f$$

By Theorem 5.1, this relation $\leq_F$ is equal to the Weak Sorted-Pareto order $\preccurlyeq_{\text{SP}}$.

Theorem 5.1 » Sorted-Pareto and $\leq_F$ result

$\leq_F$ is equal to the Weak Sorted-Pareto order $\preccurlyeq_{\text{SP}}$. ⋄

Proof: First we show that for all $\alpha, \beta \in \mathcal{A}, \alpha \preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \leq_F \beta$. Assume $\alpha \preccurlyeq_{\text{SP}} \beta$. This implies, by definition of $\preccurlyeq_{\text{SP}}$, $v(\alpha)^{\uparrow} \preccurlyeq_{\text{P}} v(\beta)^{\uparrow}$, which means that $\alpha_{(i)} \leq \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$. Therefore, for any f, $\sum_{i=1}^{m} f(\alpha_{(i)}) \leq_f \sum_{i=1}^{m} f(\beta_{(i)})$. Since $v(\alpha)^{\uparrow}$ and $v(\beta)^{\uparrow}$ are permutations of α and β respectively, then $\sum_{i=1}^{m} f(\alpha_{(i)}) \leq_f$

$\sum_{i=1}^{m} f(\beta_{(i)}) \Leftrightarrow \sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$, which implies, for any monotonic function f, $\alpha \leq_f \beta$. Since this is true for any f, then $\alpha \leq_F \beta$.

Now we show that for all $\alpha, \beta \in \mathcal{A}$, $\alpha \not\preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \not\leq_F \beta$. Assume $\alpha \not\preccurlyeq_{\text{SP}} \beta$. This implies, by definition of $\preccurlyeq_{\text{SP}}$, $v(\alpha)^{\uparrow} \not\preccurlyeq_{\text{P}} v(\beta)^{\uparrow}$. Therefore, $\exists i \in \{1, \ldots, m\}$ such that $\alpha_{(i)} \not\leq \beta_{(i)}$. We can construct a monotonic function f such that $\alpha \not\leq_f \beta$, and therefore $\alpha \not\leq_F \beta$. For instance, assign f such that $f(v) = 0$ if $v < \alpha_{(i)}$, and $f(v) = 1$ otherwise. Hence $f(\alpha_{(j)}) = 1$ for all $j \geq i$ and $f(\beta_{(j)}) = 0$ for all $j \leq i$. Therefore, $\sum_{j=1}^{m} f(\alpha_{(j)}) \geq m - i + 1$ and $\sum_{j=1}^{m} f(\beta_{(j)}) \leq m - i$, so $\sum_{j=1}^{m} f(\alpha_{(j)}) > \sum_{j=1}^{m} f(\beta_{(j)})$. Hence, $\exists f$ such that $\alpha \not\leq_f \beta$, which shows $\alpha \not\leq_F \beta$. ∎

Now we define an ordering based on the set of *strictly monotonic* weight functions. We recall the definition for a strictly monotonic function as follows.

Recall » Strictly monotonic function (Definition 2.33)
For some ordered scale T, ordered by $\leq$, a function $f : T \to \mathbb{R}^+$ is *strictly monotonic* with respect to scale T, if and only if

— $u < v \Leftrightarrow f(u) < f(v)$ for all $u, v \in T$

or equivalently, if and only if

— $u \leq v \Leftrightarrow f(u) \leq f(v)$ for all $u, v \in T$ «

Now we define the set F'_T as follows.

Definition 5.5 » Set F'_T of strictly monotonic functions
For some ordered scale T, let F'_T be the set of all possible weight functions such that $f \in F'_T$ if and only if f is *strictly monotonic* with respect to scale T. «

Given this set, we define the order relation $\leq_{F'}$ on our set of decisions $\mathcal{A}$ as follows:

Definition 5.6 » Order relation $\leq_{F'}$
For all $\alpha, \beta \in \mathcal{A}$,

— $\alpha \leq_{F'} \beta \Leftrightarrow \left[\alpha \leq_f \beta, \text{ for all } f \in F'_T\right]$ «

We also define the relation $<_{\cap_{F'}}$ as follows:

Definition 5.7 » Order relation $<_{\cap_{F'}}$
Let $<_{\cap_{F'}}$ be the intersection of all $<_f$ such that $f \in F'_T$, i.e., we have

$$- <_{\cap_{F'}} = \bigcap_{f \in F'_T} <_f$$ «

Therefore, for all $\alpha, \beta \in \mathcal{A}$, $\alpha <_{\cap_{F'}} \beta$ if and only if $\alpha <_f \beta$ for all $f \in F'_T$. Given these definitions, we have the following results.

Theorem 5.2 » Sorted Pareto and $\leq_{F'}$ result

We have that: $\preccurlyeq_{\text{SP}} = \leq_F = \leq_{F'}$. ◇

Proof: First we show that for all $\alpha, \beta \in \mathcal{A}, \alpha \preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \leq_F \beta$, and $\alpha \preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \leq_{F'} \beta$. Assume $\alpha \preccurlyeq_{\text{SP}} \beta$. This implies, by definition of $\preccurlyeq_{\text{SP}}$, that for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$. Therefore, for any f, $\sum_{i=1}^{m} f(\alpha_{(i)}) \leq \sum_{i=1}^{m} f(\beta_{(i)})$. Since $v(\alpha)^{\uparrow}$ and $v(\beta)^{\uparrow}$ are permutations of $v(\alpha)$ and $v(\beta)$ respectively, then $\sum_{i=1}^{m} f(\alpha_{(i)}) \leq \sum_{i=1}^{m} f(\beta_{(i)}) \Leftrightarrow \sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$, which implies $\alpha \leq_f \beta$. Since this is true for all f monotonic with respect to T, then $\alpha \leq_F \beta$. Also, since $F'_T \subset F_T$, i.e., the set of all strictly monotonic functions is a subset of the set of monotonic functions, then we have for all f strictly monotonic with respect to T, $\alpha \leq_f \beta$ and therefore $\alpha \leq_{F'} \beta$.

Now we show that for all $\alpha, \beta \in \mathcal{A}$, $\alpha \not\preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \not\leq_{F'} \beta$, and $\alpha \not\preccurlyeq_{\text{SP}} \beta \Rightarrow \alpha \not\leq_F \beta$. Assume $\alpha \not\preccurlyeq_{\text{SP}} \beta$. This implies, by definition of $\preccurlyeq_{\text{SP}}$, that it is not the case that for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$. Therefore, there exists $i \in \{1, \ldots, m\}$ such that $\alpha_{(i)} \not\leq \beta_{(i)}$. We can construct a strictly monotonic function f such that $\alpha \not\leq_f \beta$, and therefore $\alpha \not\leq_{F'} \beta$. For instance, assign f such that $0 \leq f(v) \leq \epsilon$ if $v < \alpha_{(i)}$, and $1 \leq f(v) \leq 1 + \epsilon$ otherwise, where ϵ is chosen suitably small (e.g., $\epsilon = 1/(m+1)$). For this choice of f, we have $f(\alpha_{(j)}) \in [0, \epsilon]$ for all $j < i$, and $f(\alpha_{(j)}) \in [1, 1+\epsilon]$ for all $j \geq i$. Therefore, $\sum_{j=1}^{m} f(\alpha_{(j)}) \geq m - i + 1$. We also have $f(\beta_{(j)}) \in [0, \epsilon]$ for all $j \leq i$, and $f(\beta_{(j)}) \in [0, 1+\epsilon]$, for all $j > i$. Therefore, $\sum_{j=1}^{m} f(\beta_{(j)}) \leq m - i + m\epsilon$. So we have $\sum_{j=1}^{m} f(\alpha_{(j)}) \geq m - i + 1$ and $\sum_{j=1}^{m} f(\beta_{(j)}) \leq m - i + m\epsilon$, and since $\epsilon < 1/m$, we have $\sum_{j=1}^{m} f(\alpha_{(j)}) > \sum_{j=1}^{m} f(\beta_{(j)})$, and therefore $\sum_{j=1}^{m} f(\alpha_j) > \sum_{j=1}^{m} f(\beta_j)$, i.e., $\alpha \not\leq_f \beta$. Hence, there exists f that is strictly monotonic with respect to T such that $\alpha \not\leq_f \beta$, so we have $\alpha \not\leq_{F'} \beta$. Also, since f is strictly monotonic with respect to T, then *a fortiori* f is monotonic with respect to T, and we have $\alpha \not\leq_F \beta$. ∎

Corollary 5.3 » Sorted-Pareto and $<_{\cap_{F'}}$ result

We have that: $\prec_{\text{SP}} = <_{\cap_{F'}}$ ◇

Proof: First we show that for all $\alpha, \beta \in \mathcal{A}$, $\alpha <_{\cap_{F'}} \beta \Rightarrow \alpha \prec_{\text{SP}} \beta$. Suppose $\alpha <_{\cap_{F'}} \beta$. Then, for all $f \in F'$, $\alpha <_f \beta$, and so for all $f \in F'$, $\alpha \leq_f \beta$ and hence $\alpha \leq_{F'} \beta$. Thus by Theorem 5.2 we have that $\alpha \preccurlyeq_{\text{SP}} \beta$. This means that either $\alpha \prec_{\text{SP}} \beta$, or $\alpha \equiv_{\text{SP}} \beta$. Assume $\alpha \equiv_{\text{SP}} \beta$. Then $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$, i.e., $\alpha_{(i)} = \beta_{(i)}$, for all $i \in \{1, \ldots, m\}$, and therefore we could not have any $f \in F'$ such that $\alpha <_f \beta$. This contradicts $\alpha <_{\cap_{F'}} \beta$, so we must have that $\alpha \prec_{\text{SP}} \beta$.

Now we show that for all $\alpha, \beta \in \mathcal{A}$, $\alpha \prec_{\text{SP}} \beta \Rightarrow \alpha <_{\cap_{F'}} \beta$. Suppose $\alpha \prec_{\text{SP}} \beta$, then by definition of $\prec_{\text{SP}}$, we have $\alpha \preccurlyeq_{\text{SP}} \beta$. Thus, from Theorem 5.2, we have $\alpha \leq_{F'} \beta$, i.e., for all $f \in F'$, $\alpha \leq_f \beta$. We need to show that for all $f \in F'$, $\alpha <_f \beta$, i.e., for all $f \in F'$, $\beta \not\leq_f \alpha$. Assume for some $f \in F'$, $\beta \leq_f \alpha$. Since $\alpha \leq_f \beta$ and $\beta \leq_f \alpha$, we have $\sum_{i=1}^{m} f(\alpha_i) = \sum_{i=1}^{m} f(\beta_i)$ and therefore $\sum_{i=1}^{m} f(\alpha_{(i)}) = \sum_{i=1}^{m} f(\beta_{(i)})$. Also, since we have $\alpha \preccurlyeq_{\text{SP}} \beta$, we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} \leq \beta_{(i)}$, and since $\sum_{i=1}^{m} f(\alpha_{(i)}) = \sum_{i=1}^{m} f(\beta_{(i)})$ we have for all $i \in \{1, \ldots, m\}$, $\alpha_{(i)} = \beta_{(i)}$. This contradicts $\alpha \prec_{\text{SP}} \beta$, therefore, we have that $\alpha <_f \beta$. Since this is true for all $f \in F'$, then we have that $\alpha <_{\cap_{F'}} \beta$. ∎

This gives a semantics to Sorted-Pareto, as a relation that can be used in decision making situations where there may only be ordinal or qualitative information available, and it provides an ordering that is consistent with any possible weights function selected to map an ordinal scale to a numerical one. It can be viewed as a more cautious representation than a weighted constraints one, and it can be applied in many of the application areas of weighted constraints. The weighted constraints formalism assumes that the costs are on an additive scale, where the cost of A and B is the sum of the costs of A and B; however, in many situations this can be questionable. Different experiments have shown [Lar92, LM95] that the elicitation of numerical preferences from experts can be difficult. For example, for a Bayesian network [Pea88], one can map the network to a soft constraint problem where the constraints correspond to conditional probability tables, and then solving for an optimal solution using a weighted constraints solver is the equivalent of *most probable explanation* (MPE) task [FL93]. Suppose one is using a weighted constraints solver to find a most probable explanation. Since the elicitation of probabilities can be

problematic and unreliable, then instead of taking the elicited values at face value, one considers them as just representing the ordering between the probabilities. In this case, Theorems 5.1 and 5.2 shows that Sorted-Pareto represents the order relation that all compatible probability assignments agree with.

5.4 Soft Constraints and Sorted-Pareto Dominance

As discussed in Section 3.1, *soft constraints* can be used to model many real-world problems when there is a need to specify preferences on particular aspects of the problem solutions. A soft constraint associates a preference degree to an assignment of a set of decision or problem variables. These preference degrees associated with an assignment can be combined to give the overall preference level of the assignment, usually with the aim of ordering these assignments and to obtain a set of optimal solutions. In this section, we look at an instance of a constraints problem for Sorted-Pareto dominance.

First we revisit our definitions of a preference degree structure (PDS) and a general constraints problem (GCP), as given in Section 3.4.

Recall » Preference degree structure (Definition 3.11)
A *preference degree structure* (PDS) is a tuple $\langle I, \otimes, \preceq \rangle$, where

- I is a set of preference degrees;
- $\otimes$ is a commutative and associative operator, monotonic with respect to $\preceq$ (i.e., for $a, b, c \in I$, $a \preceq b \Rightarrow a \otimes c \preceq b \otimes c$), which is used to combine the preference degrees;
- $\preceq$ is a preorder relation on the set of degrees I. «

Recall » General constraint problem (Definition 3.12)
A *general constraints problem* (GCP) is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where

- $\mathcal{X}$ is a set of n variables, $\{X_1, \ldots, X_n\}$;
- $\mathcal{D}$ is a set of variable domains, $\{\mathcal{D}(X_1), \ldots, \mathcal{D}(X_n)\}$;
- $\mathcal{C}_H$ is a set of hard constraints;
- $\mathcal{C}_S$ is a multiset of soft constraints, where for all $s_V \in \mathcal{C}_S$, with scope $V \subseteq \mathcal{X}$, $s_V : \mathcal{D}(V) \to I$;
- $\mathcal{P}$ is a preference degree structure $\langle I, \otimes, \preceq \rangle$. «

For a general constraints problem, we recall that $\text{Sol}(P)$ is the set of all complete assignments that are consistent with the hard constraints in $\mathcal{C}_H$. The preference level $\rho(t)$ of an assignment $t \in \text{Sol}(P)$ is the combination of all the preference degrees associated to t by the soft constraints in $\mathcal{C}_S$, i.e., $\rho(t) = \otimes_{s_V \in \mathcal{C}_S} s_V(t)$. An optimal solution $t \in \text{Sol}(P)$ is one such that there is no other $t' \in \text{Sol}(P)$ such that $\rho(t') \preccurlyeq \rho(t)$. Now we give the definition for a Sorted-Pareto constraints problem as follows.

Definition 5.8 » Sorted-Pareto constraints problem (SPCP)
A Sorted-Pareto constraints problem is a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$ where the preference degree structure $\mathcal{P} = \langle I, \otimes, \preceq \rangle$ is given by:

- $I = M^T$, i.e., the set of preference degrees is the set of multisets of T,
- $\otimes = \uplus$, i.e., the combination operator is *multiset sum*,
- $\preceq = \preccurlyeq_{\text{SP}}$, i.e., the order relation is the induced Sorted-Pareto preference relation defined on M^T, where for $A, B \in M^T$, $A \preccurlyeq_{\text{SP}} B \iff A^{\uparrow} \leq B^{\uparrow}$ «

The preference level $\rho(t)$ of a solution $t \in \text{Sol}(P)$ for a Sorted-Pareto constraints problem is defined as follows.

Definition 5.9 » Preference level (SPCP)
Given a Sorted-Pareto constraints problem, the preference level $\rho(t)$ of a solution $t \in \text{Sol}(P)$ is given by

— $\rho(t) = \biguplus_{s_V \in \mathcal{C}_S} s_V(t) = \{s_V(t) : s_V \in \mathcal{C}_S\}$ «

i.e., it is the multiset of all the preference degrees associated to the solution by the soft constraints of the problem.

We can now define the Sorted-Pareto dominance relations over constraints problems.

Definition 5.10 » Weak Sorted-Pareto Dominance (SPCP)
For all $t, t' \in \text{Sol}(P)$, t *Weak Sorted-Pareto dominates* t', written as $t \preccurlyeq_{\text{SP}} t'$, if and only if

— $\rho(t) \preccurlyeq_{\text{SP}} \rho(t')$ «

That is, a solution t Weak Sorted-Pareto dominates another solution t' if the multiset of preference values associated to t Weak Sorted-Pareto dominates the multiset of

preference values associated to t'.

Definition 5.11 » Sorted-Pareto Dominance (SPCP)
For all $t, t' \in \text{Sol}(P)$, t *Sorted-Pareto dominates* t', written as $t \prec_{\text{SP}} t'$, if and only if

— $\rho(t) \prec_{\text{SP}} \rho(t')$ «

That is, a solution t Sorted-Pareto dominates another solution t' if the multiset of preference values associated to t Sorted-Pareto dominates the multiset of preference values associated to t'.

Definition 5.12 » Sorted-Pareto Equivalence (SPCP)
For all $t, t' \in \text{Sol}(P)$, t is *Sorted-Pareto equivalent* to t', written as $t \equiv_{\text{SP}} t'$, if and only if

— $\rho(t) \equiv_{\text{SP}} \rho(t')$, or equivalently, $\rho(t) = \rho(t')$ «

That is, a solution t is Sorted-Pareto equivalent to another solution t' if the multiset of preference values associated to t is equal to the multiset of preference values associated to t'. An optimal solution of a Sorted-Pareto constraints problem is defined as follows.

Definition 5.13 » Optimal solution (SPCP)
Given a Sorted-Pareto constraints problem P, an solution $t \in \text{Sol}(P)$ is *optimal* if and only if

— there exists no $t' \in \text{Sol}(P)$ such that $t' \prec_{\text{SP}} t$ «

Let us look at an example.

Example 5.2 ► Sorted-Pareto constraints problem example**.**
Let us consider a Sorted-Pareto constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$*, where we have three variables,* X_1*,* X_2 *and* X_3*, and where the domains of the variables are* $\mathcal{D}(X_1) = \{a, b\}$*,* $\mathcal{D}(X_2) = \{c, d\}$ *and* $\mathcal{D}(X_3) = \{e, f\}$*.*

The problem has two hard constraints c_{13} *and* c_{23} *such that* $\bar{c}_{13} = \{(a, e)\}$ *and* $\bar{c}_{23} = \{(c, e)\}$*, which specify the tuples that are not allowed by the constraints.*

The problem has two soft constraints s_{12} *and* s_{13}*, which are defined in the following tables.*

s_{12}		X_1: a	X_1: b
X_2	c	4	6
	d	3	1

s_{13}		X_1: a	X_1: b
X_3	e	4	5
	f	2	6

The next table shows the tuples in $\mathcal{D}(\mathcal{X})$ and the preference level of each solution, which is given by the multiset of the values associated to the solution by the soft constraints of the problem. Not all tuples in $\mathcal{D}(\mathcal{X})$ are solutions of $\mathrm{Sol}(P)$*, since they do not satisfy the hard constraints.*

	t	$\rho(t)$
t_1	(a,c,e)	n/a
t_2	(a,c,f)	$\{2,4\}$
t_3	(a,d,e)	n/a
t_4	(a,d,f)	$\{2,3\}$
t_5	(b,c,e)	n/a
t_6	(b,c,f)	$\{6,6\}$
t_7	(b,d,e)	$\{1,5\}$
t_8	(b,d,f)	$\{1,6\}$

We can see that the Sorted-Pareto optimal solutions are t_4 and t_7, since any other solution is strictly dominated by either t_4 or t_7 and we also have that $t_4 \not\succ_{\mathrm{SP}} t_7$ and $t_7 \not\succ_{\mathrm{SP}} t_4$.

Therefore, we have the Sorted-Pareto optimal set

- $\mathrm{O}_{\mathrm{SP}} = \{t_4, t_7\}$ ▲

5.5 Solving Constraints Problems

In this section we describe some algorithms for solving a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$. First we describe the algorithms in a general context and then we also discuss some specific algorithmic details relating to Sorted-Pareto constraints problems. We detail three different algorithms: the first is a brute force search, the second is a depth first branch and bound search which uses a lower bound, and the third is a depth first brand and bound algorithm which uses both a lower and upper bound. We also look at variations of these three algorithms where equivalences between solutions are explicitly handled.

5.5.1 *BruteSearch* algorithm

In this section, we look at the algorithm which is outlined in Figure 5.1. In this depth first search algorithm, each problem variable in turn is chosen and assigned a value from its domain. If the algorithm encounters a variable assignment that is not consistent with the hard constraints in the problem, then the search will backtrack, as in the backtracking search detailed in Section 3.5.1. Also, the domains of the variables are updated and inconsistent values are removed, in order to maintain some form of consistency with the set of hard constraints, as done in the look-ahead search in Section 3.5.1. If a variable has no legal values left in its domain, then the search will backtrack. The algorithm also maintains a set of non-dominated or optimal assignments, and once a complete consistent assignment is encountered, the algorithm will compare the preference level of this assignment (given by the soft constraints) with any previously found non-dominated complete assignments, and update the set of non-dominated assignments if this new complete assignment is non-dominated. Once the procedure is finished, the output of the algorithm is the set of non-dominated or optimal solutions to the problem. This algorithm, as described, is a generate and test (or "brute force") approach to the soft constraints as detailed in Section 3.5.2, so we label this algorithm *BruteSearch*.

Algorithm 5.1: *BruteSearch* recursive algorithm for generating a set of optimal solutions to a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$

```
   Input  : GCP problem P, partial assignment t
   Output : set of optimal solutions O_P
 1 begin
 2     if HasNextVar() = false then
 3         if IsStrictlyDominated(t, O_P) = false then
 4             O_P ← t ∪ RemoveStrictlyDominated(O_P, {t})
 5     else
 6         X ← NextVar()
 7         while HasNextVal(X) do
 8             t ← t ∪ (X, NextVal(X))
 9             if IsConsistent(t) then
10                 return BruteSearch(t, O_P)
11     return O_P
```

The auxiliary functions used in the *BruteSearch* algorithm in Figure 5.1 are described as follows.

HasNextVar(P, t)

Given problem P and assignment t, this function returns **true** if t is not a complete assignment.

NextVar(P, t)

Given problem P and assignment t, this function returns the next unassigned variable X that is to be assigned by t.

HasNextVal(X)

Given variable X, this function returns **true** if there is a value remaining in the domain of X.

NextVal(X)

Given variable X, this function returns the next value a from the domain of variable X.

IsStrictlyDominated(t, S)

Given set of complete assignments S, the function returns **true** if there exists $s \in S$ such that $\rho(t)$ is strictly dominated by $\rho(s)$.

RemoveStrictlyDominated(S, S')

Given sets of complete assignments S and S', this function returns a new set containing all $s \in S$ such that the preference level of s is not dominated by the preference level of any $s' \in S'$.

IsConsistent(t)

Given assignment t, this function returns **true** if t is consistent with all hard constraints $c_V \in \mathcal{C}_H$, given the consistency algorithm being used by the solving process.

Let us look at an example. As in Section 3.5, for presentation purposes we assume a fixed variable ordering $(X_1, \ldots, X_n)$.

Example 5.3 ► *BruteSearch* example**.**
Figure 5.1 shows the search tree for the BruteSearch algorithm for a problem with three variables, X_1*,* X_2 *and* X_3*, and where the original domains of the variables are* $\mathcal{D}(X_1) = \{a, b\}$*,* $\mathcal{D}(X_2) = \{c, d\}$ *and* $\mathcal{D}(X_3) = \{e, f\}$*.*

The problem has two hard constraints, c_{13} *and* c_{23} *such that* $\bar{c}_{13} = \{(a, e)\}$ *and* $\bar{c}_{23} = \{(c, e)\}$*, which specify the tuples that are not allowed by the constraints.*

The problem has two soft constraints s_{12} *and* s_{13}*, defined in the following tables.*

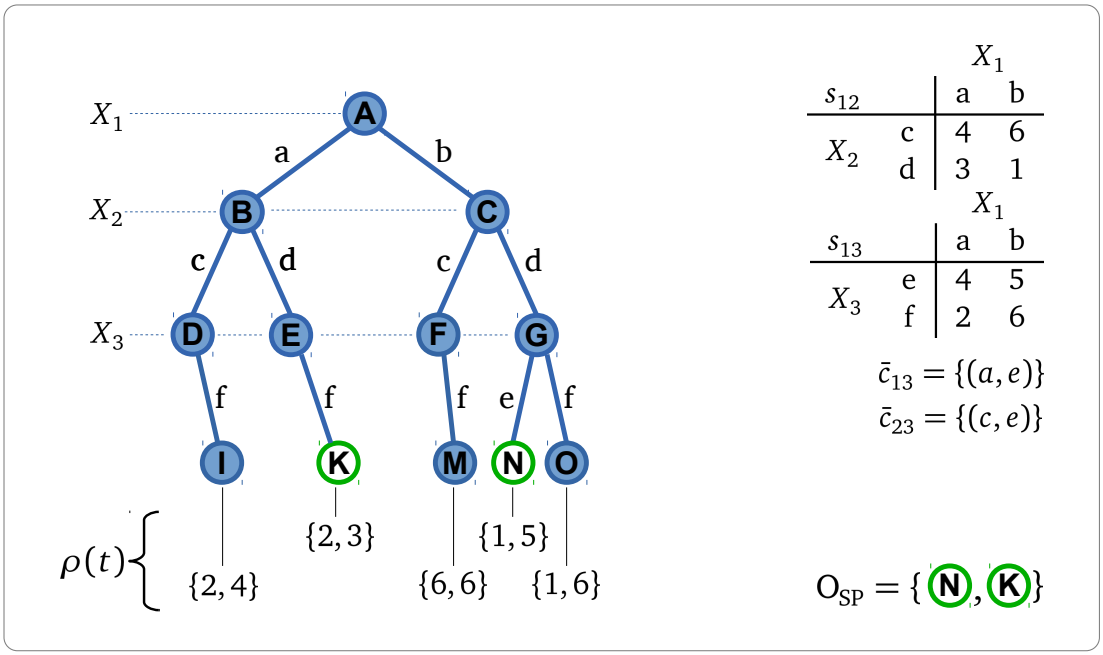


Figure 5.1: For Example 5.3, *BruteSearch* search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$, with two hard and two soft constraints as given in the figure. The *BruteSearch* checks the preference level of complete assignments with any previously found optimal solutions.

s_{12}		X_1: a	b
X_2	c	4	6
	d	3	1

s_{13}		X_1: a	b
X_3	e	4	5
	f	2	6

The BruteSearch algorithm removes any inconsistent values from the variable domains. Once a complete consistent assignment is encountered, the algorithm will compare the preference level of this assignment with any previously found non-dominated complete assignments.

For example, at node M*, the preference level of the tuple associated with node* M *is compared with the preference level of the tuple associated with node* K*, which is the current optimal assignment at that point in the search. At node* O*, the preference level of the tuple associated with node* M *is compared with nodes* K *and* N*, since these are both optimal at that point in the search.* ▲

5.5.2 *DFBBSearch* algorithm

In this section we look at the algorithm outlined in Figure 5.2. In this depth first branch and bound (DFBB) search, as detailed in Section 3.5.2, instead of only testing

if complete assignments are dominated by the set of non-dominated solutions, the algorithm will also check to see if partial assignments are dominated. It does this as follows. First it calculates a lower bound for the current partial assignment, which is a lower bound on the preference level of any complete assignment extending this partial assignment. Then, if this bound is strictly dominated by some previously found non-dominated solution, the search will backtrack, since all completed assignments extending the current partial assignment will be dominated. As in the *BruteSearch* algorithm, once the search is finished, the output of the algorithm is the set of non-dominated or optimal solutions to the problem. This approach improves on the *BruteSearch* by eliminating parts of the search space that do not contain any non-dominated solutions, therefore eliminating unnecessary dominance checking and reducing the amount of time the algorithm spent doing these checks. We label this algorithm *DFBBSearch*.

Algorithm 5.2: *DFBBSearch* recursive algorithm for generating a set of optimal solutions to a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$

```
   Input  : problem P, partial assignment t
   Output : set of optimal solutions O_P
 1 begin
 2     if HasNextVar() = false then
 3         if IsStrictlyDominated(t, O_P) = false then
 4             O_P ← t ∪ RemoveStrictlyDominated(O_P, {t})
 5         return O_P
 6     X ← NextVar()
 7     while HasNextVal(X) do
 8         t ← t ∪ (X, NextVal(X))
 9         lb ← CalculateLowerBound(t)
10         if IsConsistent(t) ∧ ¬IsBoundDominated(lb, O_P) then
11             return DFBBSearch(t, O_P)
12     return O_P
```

The additional auxiliary functions used in the *DFBBSearch* algorithm in Figure 5.2 are described as follows.

IsBoundDominated(lb, S)

Given a lower bound preference level lb, and a set of complete assignments S this function returns **true** if there exists $s \in S$ such that lb is dominated by $\rho(s)$.

CalculateLowerBound(t)

Given (partial) assignment t, this function calculates a lower bound prefer-

ence level for t.

Calculating lower bound for Sorted-Pareto. For the Sorted-Pareto constraints problem, we can calculate a lower bound preference level for some partial assignment t, as follows.

Definition 5.14 » Lower bound for Sorted-Pareto constraints problem
For a Sorted-Pareto constraints problem, for partial assignment t, let $s_1, \ldots, s_m$ be the soft constraints in $\mathcal{C}_S$ once t has been instantiated. Then the lower bound preference level $\rho_*(t)$ of t is given as

— $\rho_*(t) = \{s_1^{\min}, \ldots, s_m^{\min}\}$, where $s_i^{\min} = \min_{u \in scope(s_i)} s_i(u)$ «

That is, $s_i^{\min}$ is the minimum value that s_i can take over all tuples $u \in \mathcal{D}(\mathcal{X})$, where u is scoped to s_i. Let us look at an example.

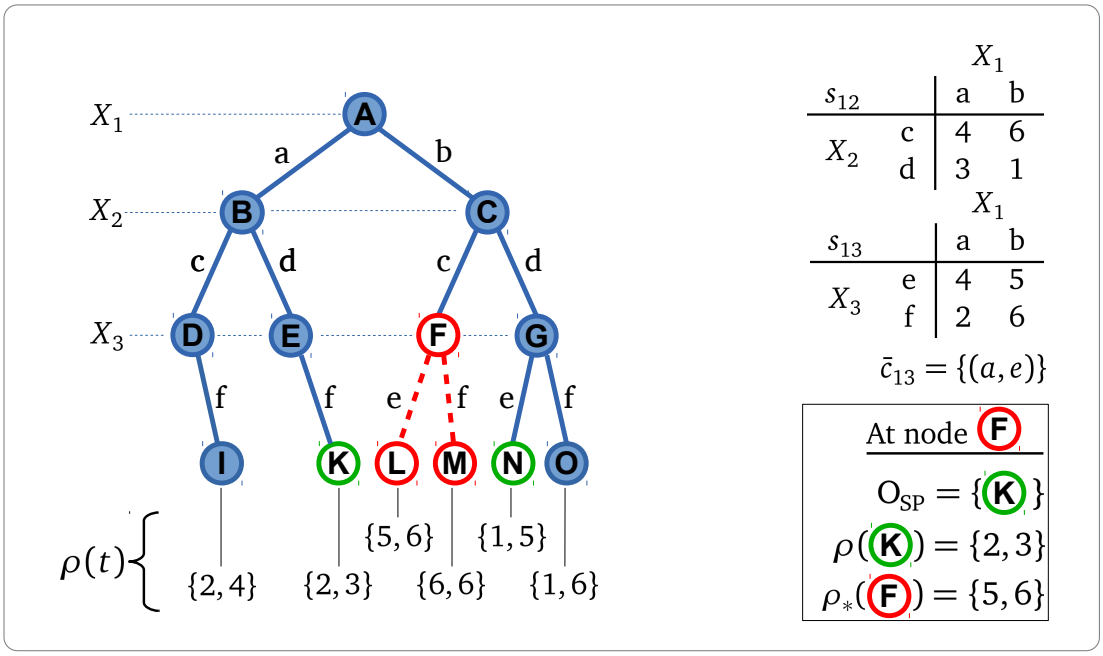


Figure 5.2: For Example 5.4, *DFBBSearch* search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$, with one hard and two soft constraints as given in the figure. The *DFBBSearch* generates a lower bound and will backtrack if the bound is dominated by any previously found optimal solutions.

Example 5.4 ► Lower bound for Sorted-Pareto example.
Figure 5.2 shows the search tree for the DFBBSearch algorithm for a problem with

three variables, X_1, X_2 and X_3, and where the original domains of the variables are $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$.

This time the problem has one hard constraint c_{13}, such that $\bar{c}_{13} = \{(a, e)\}$, which specifies the tuples that are not allowed by the constraint, and two soft constraints s_{12} and s_{13}, defined in the following tables.

s_{12}		X_1: a	X_1: b
X_2	c	4	6
	d	3	1

s_{13}		X_1: a	X_1: b
X_3	e	4	5
	f	2	6

The BruteSearch algorithm calculates a lower bound for the current partial assignment, which is a lower bound on the preference level of any complete assignment extending this partial assignment. If this bound is strictly dominated by some previously found non-dominated solution, the search will backtrack.

For example, at node (F), the lower bound preference of the assignment represented by node (F) is given by

- $\rho_*((b, c)) = \{s_{12}^{\min}, s_{13}^{\min}\} = \{5, 6\}$.

Since this lower bound is strictly dominated by the preference level of current non-dominated solution represented by (K), i.e.,

- $\rho((a, d, f)) = \{2, 3\}$,

then the BruteSearch algorithm will backtrack at (F). ▲

5.5.3 *PANDSearch* algorithm

In this section we look at another algorithm which is outlined in Figure 5.3. Since in this problem we are dealing with a partial order on the set of complete assignments, we have a set of non-dominated solutions at each point in the search. However, not all of these assignments are relevant in each part of the search space, so if it can be shown that some complete assignment s fails to dominate any complete assignment extending partial assignment t, then there is no need to consider assignment s in the search space extending below t. To determine which previously found complete assignments are relevant to a partial assignment at a particular point in the search, the algorithm calculates an upper bound for the current partial assignment, which is a upper bound on the preference level of any complete assignment extending

the current assignment. If this bound is not strictly dominated by the previously found complete assignment s, then s can be ignored in the subsearch extending below t. This potentially improves on the previous algorithm by further eliminating unnecessary dominance checks, but at the cost of performing this extra test. A similar idea in [WT11] has been shown to be effective in optimisation with respect to comparative preferences.

Algorithm 5.3: *PANDSearch* recursive algorithm for generating a set of optimal solutions to a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$

Input : problem P, partial assignment t, undominated solution set RUS
Output : pair of solution sets $\langle \text{RUS}, \text{NEW} \rangle$

```
1  begin
2      NEW ← ∅
3      if HasNextVar() = false then
4          if IsStrictlyDominated(t, RUS) = false then
5              NEW ← {t}
6              RUS ← RemoveStrictlyDominated(RUS, NEW)
7          return ⟨RUS, NEW⟩
8      RUS′ ← RUS
9      X ← NextVar()
10     while HasNextVal(X) do
11         t ← t ∪ (X, NextVal(X))
12         lb ← CalculateLowerBound(t)
13         if IsConsistent(t) ∧ ¬IsBoundDominated(lb, RUS′) then
14             ub ← CalculateUpperBound(t)
15             ⟨RUS′, OTHERS⟩ ← Partition(ub, RUS′)
16             ⟨RUS′, NEW′⟩ ← PANDSearch(t, RUS′)
17             OTHERS ← RemoveStrictlyDominated(OTHERS, NEW′)
18             RUS′ ← RUS′ ∪ OTHERS ∪ NEW′
19     NEW ← RUS′ \ RUS
20     RUS ← RUS′ ∩ RUS
21     return ⟨RUS, NEW⟩
```

The main details of the algorithm are outlined as follows. The recursive function *PANDSearch* takes as input a partial assignment t and a set RUS, which is the set of relevant undominated solutions inherited from the parent node. If t is a complete assignment, i.e., if there are no more variables to be assigned (line 3), then, if t is not dominated by any solution in RUS (line 4), it is added to the set NEW (line 5). Any solutions in RUS that are dominated by t are removed (line 6), and the pair of sets RUS and NEW are returned (line 7). Otherwise, if t is not a complete assignment, then a variable X is chosen (line 9), and a value in the domain of X is

chosen to extend partial assignment t (line 11). A lower bound preference level ub for t is calculated (line 12), and if the lower bound of t is not dominated by any other solution (line 13), then the search continues. An upper bound preference level ub for t is calculated (line 14), and any previously found complete assignment s that does not dominate this upper bound is removed from RUS, and added to set variable OTHERS to allow such s to be restored on backtracking (line 15). The search continues with the recursive call to *PANDSearch* (line 16), until all non-dominated solutions are found and returned (line 21). The output of the algorithm is a pair of sets, and NEW contains the set of non-dominated solutions to the problem.

The additional auxiliary functions used in the *PANDSearch* algorithm in Figure 5.3 are described as follows.

CalculateUpperBound(t)
: Given (partial) assignment t, this function calculates an upper bound preference level for t.

Partition(ub, S)
: Given an upper bound preference level ub and a set of complete assignments S, this function returns a pair of sets $\langle S', S'' \rangle$, where S' contains the elements of S whose preference levels strictly dominate the bound ub, and S'' contains the elements of S whose preference levels do not strictly dominate the bound ub, i.e., $S' = \{s \in S : \rho(s) \prec ub\}$, and $S'' = \{s \in S : \rho(s) \not\prec ub\}$.

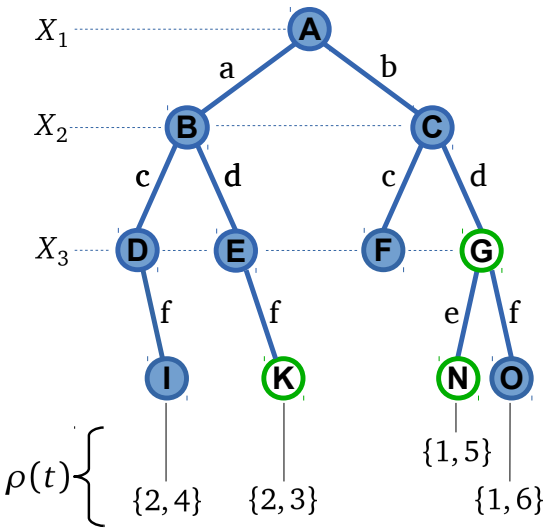
Calculating upper bound for Sorted-Pareto. For the Sorted-Pareto constraints problem, we can calculate an upper bound preference level for some partial assignment t, as follows.

Definition 5.15 » Upper bound for Sorted-Pareto constraints problem
For a Sorted-Pareto constraints problem, for partial assignment t, let $s_1, \ldots, s_m$ be the soft constraints in $\mathcal{C}_S$ once t has been instantiated. Then the upper bound preference level $\rho^*(t)$ of t is given as

— $\rho^*(t) = \{s_1^{\max}, \ldots, s_m^{\max}\}$, where $s_i^{\max} = \max_{u \in scope(s_i)} s_i(u)$ «

That is, $s_i^{\max}$ is the maximum value that s_i can take over all tuples $u \in \mathcal{D}(\mathcal{X})$, where u is scoped to s_i. Let us look at an example.

Example 5.5 ► Upper bound for Sorted-Pareto example**.**
Figure 5.3 shows the search tree for the DFBBSearch algorithm for a problem with



s_{12}		X_1 a	X_1 b
X_2	c	4	6
	d	3	1

s_{13}		X_1 a	X_1 b
X_3	e	4	5
	f	2	6

$\bar{c}_{13} = \{(a, e)\}$

At node G

$O_{SP} = \{K\}$

$\rho(K) = \{2, 3\}$

$\rho^*(G) = \{1, 6\}$

Figure 5.3: For Example 5.5, *PANDSearch* search tree for a problem with three variables X_1, X_2 and X_3, where $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$, with one hard and two soft constraints as given in the figure. The *PANDSearch* generates a upper bound to reduce the amount of optimal solutions to test against when extending a partial assignment.

three variables, X_1, X_2 and X_3, and where the original domains of the variables are $\mathcal{D}(X_1) = \{a, b\}$, $\mathcal{D}(X_2) = \{c, d\}$ and $\mathcal{D}(X_3) = \{e, f\}$.

This time the problem has one hard constraint c_{13}, such that $\bar{c}_{13} = \{(a, e)\}$, which specifies the tuples not allowed by the constraint.

The problem has two soft constraints s_{12} and s_{13}, defined in the following tables.

s_{12}		X_1 a	X_1 b
X_2	c	4	6
	d	3	1

s_{13}		X_1 a	X_1 b
X_3	e	4	5
	f	2	6

At any given node, the PANDSearch algorithm calculates an upper bound on the preference level of any complete assignment extending the assignment at that node. If this upper bound is not strictly dominated by a previously found solution, then that solution can be ignored in the dominance checks in the search below the given node.

For example, in Figure 5.3 at node G, the upper bound preference of the assignment represented by node G is given by

- $\rho^*((b, c)) = \{s_{12}^{\min}, s_{13}^{\min}\} = \{1, 6\}$.

Since this upper bound is not strictly dominated by the preference level of current non-dominated solution represented by (K)*, i.e.,*

- $\rho((a,d,f)) = \{2,3\}$,

then the PANDSearch algorithm will not consider the solution represented by node (K) *in the search below node* (G)*.* ▲

5.5.4 Handling Equivalences

We now discuss some variations of the three algorithms already detailed, in which we look at explicitly handling Sorted-Pareto equivalent solutions. Since Sorted-Pareto is a *preorder* on a set of solutions, we can have solutions that are Sorted-Pareto equivalent, i.e., for $t, t' \in \text{Sol}(P)$, we have that $\rho(t) = \rho(t')$. Given this, we define the notion of Sorted-Pareto equivalence classes (see Definitions 2.22 and 5.12) for our Sorted-Pareto constraints problem, as follows.

Definition 5.16 » Sorted-Pareto equivalence class (GCP)
Given a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *Sorted-Pareto equivalence class* of a solution $t \in \text{Sol}(P)$, denoted by $[t]_{\text{SP}}$, is defined as

— $[t]_{\text{SP}} = \{t' : t' \in \text{Sol}(P), t' \equiv_{\text{SP}} t\}$ «

Therefore, the resulting set of Sorted-Pareto non-dominated solutions to a problem can be partitioned into equivalence classes of solutions where each class contains the solutions that have the same preference level.

When searching for a set of optimal solutions, given that some solutions may have the same preference level, we look at a variation of each of the three algorithms *BruteSearch*, *DFBBSearch* and *PANDSearch* which takes equivalence into consideration. Instead of maintaining a set of undominated solutions during the search as in the original algorithms, the equivalence handling version of the algorithm maintains an undominated preference map, where each preference level in the map maps to the equivalence class corresponding to that preference level, i.e, each preference level maps to a set of undominated, equivalent solutions.

In the original algorithms, at various points during the search, the algorithms perform a dominance check of the preference level of the current solution (or in the case of the branch and bound algorithms, the lower or upper bound preference

level of the solution) against the set of previously found solutions. In the equivalence handling algorithms, instead of comparing against the preference levels of all undominated solutions (some of which may be Sorted-Pareto equivalent and have the same preference level) we compare against the preference levels in the undominated preference map. This results in a smaller amount of dominance checks and potentially saves on time spent performing these checks, however this is at the expense of maintaining a map of preference levels to solutions rather than maintaining a set of solutions.

We label the original versions of the *BruteSearch*, *DFBBSearch* and *PANDSearch* algorithms, i.e., the set-based implementations, as *Brute-A*, *DFBB-A* and *PAND-A* respectively. We label the equivalence handling versions of the algorithms as *Brute-B*, *DFBB-B* and *PAND-B*.

5.6 Implementation Details

In this section, we detail the implementation of the solver used to solve the problem instances for our experimental results in Section 5.7. The *BruteSearch*, *DFBBSearch* and *PANDSearch* algorithms, as detailed in the previous section, along with the Sorted-Pareto dominance preference relation were implemented in our own soft constraint solver. We detail some of the other algorithms and heuristics used in the solver implementation, and we also describe the problem generation process for the random problems used in the experimental results.

5.6.1 Algorithm Implementation

Maintaining consistency

To implement the *IsConsistent* function, i.e., to test during the search if an assignment t maintains some level of consistency with the hard constraints in the problem, the solver uses a MAC3 algorithm [Mac77, RvBW06, Ch. 3]. As detailed in Section 3.5.1, the algorithm revises the domains of unassigned variables if they are not consistent with the hard constraints of the problem, so that only values that are consistent remain.

Variable selection

To implement the *NextVar* function, i.e., to select the next variable to assign to some assignment t during the search, we use the *min domain over degree* heuristic [BR96]. First we define the *degree* of a variable.

Definition 5.17 » Variable degree
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *degree* of a variable $X \in \mathcal{X}$ is given by

— $\deg(X) = |\{c_V : c_V \in \mathcal{C}_H \cup \mathcal{C}_S \wedge V \ni X|\}$ «

That is, the degree of a variable is the number of hard and soft constraints for which the variable appears in the scope of the constraint. The *NextVar* function therefore chooses a variable X from $\mathcal{X}$ such that

— $X \in \arg\min_{X \in \mathcal{X}} \frac{|\mathcal{D}(X)|}{\deg(X)}$

Value selection

To implement the *NextVal* function, i.e., to select the next value from the domain of a given variable, we use a min sum of weights heuristic. Firstly, since the preference scale T is a qualitative scale, then the values in T are mapped in a canonical way to some numerical values to perform the sum of weights. We use some additional notation defined as follows. For some assignment $t \in \mathcal{D}(W)$, and some variable $X \in \mathcal{X}$, where $X \notin W$, let $t_{(X,v)}$ denote the extension of tuple t to include the assignment of domain value v to variable X, i.e, $t_{(X,v)} = t \cup (X, v)$. For some partial assignment t, let $\mathrm{T}_s(t)$ be the set of soft constraints that are completely assigned by tuple t and let $\mathrm{P}_s(t)$ be the set of soft constraints that are partially assigned by tuple t. Now we define the domain value *sum of weights* as follows.

Definition 5.18 » Domain value sum of weights
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where $t \in \mathcal{D}(W)$ and $W \subset \mathcal{X}$, then for some variable $X \in \mathcal{X}$, the *sum of weights* of a domain value $v \in \mathcal{D}(X)$, denoted by $w(v)$ is given by:

— $w(v) = \sum \{s_V(t_{(X,v)}) : s_V \in \mathrm{P}_s(t) \cap \mathrm{T}_s(t_{(X,v)})\}$ «

That is, it is the sum of the preference values associated with $t_{(X,v)}$ by the soft constraints that are completely assigned by $t_{(X,v)}$.

Then the *NextVal* function, using the min sum of weights heuristic, chooses a domain value v' from $\mathcal{D}(X)$ such that:

— $v' \in \arg\min_{v \in \mathcal{D}(X)} w(v)$

5.6.2 Problem Generator Implementation

For the purposes of performing the experiments, we developed a constraint problem generator to generate some random problem instances. In this section, we describe the generation process we used to create the random problems and also describe the parameters to the generator. In the problem instances generated by our constraint generator, the size of the scope of each constraint is two, i.e., binary constraints, and therefore the following definitions and descriptions are in terms of binary constraints.

First we define some properties of the random constraint problems, which form parameters to the constraint problem generator.

Definition 5.19 » Normalised problem
We say that a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$ is *normalised* if there are no two hard constraints $c_V, c_W \in \mathcal{C}_H$ such that $V = W$. «

That is, there are no two hard constraints in the problem that have the same scope. We now define problem size and domain size.

Definition 5.20 » Problem size (n)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *problem size* is given by

— $n = |\mathcal{X}|$ «

That is, the problem size is the number of variables in the problem.

Definition 5.21 » Domain size (m)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *domain size* of a variable $X \in \mathcal{X}$ is given by

— $m(X) = |\mathcal{D}(X)|$ «

For a problem instance where all the variables in the problem have the same size domain, we use m to denote this size. Given these parameters, the *solution space* of a problem is defined as follows.

Definition 5.22 » Solution space (ss)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *solution space* is given by

— $ss = |\mathcal{D}(\mathcal{X})| = m(X_1) \times \cdots \times m(X_n) = m^n$ «

We now look at some properties of the hard constraints in a given problem. We define the hard constraint count and hard constraint density.

Definition 5.23 » Hard constraint count (hc)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *hard constraint count* is given by

— $hc = |\mathcal{C}_H|$ «

That is, hc denotes the number of hard constraints in the problem.

Definition 5.24 » Hard constraint density (hd)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *hard constraint density* is given by

— $hd = \dfrac{hc}{(n * (n-1)/2)}$ «

That is, the hard constraint density is the fraction of hard constraints in the problem with respect to the maximum number of constraints in a normalised problem with binary hard constraints. Since we consider only normalised problems with binary hard constraints, we can easily derive the hard constraint count from the hard constraint density and vice versa. Now we define the tightness property of a hard constraint.

Definition 5.25 » Hard constraint tightness (ht)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, for a hard binary constraint $c_V \in \mathcal{C}_H$, the *constraint tightness* is given by

— $ht(c_V) = \dfrac{|\{t \in \mathcal{D}(\mathcal{X}) : t \notin c_V\}|}{m^2}$ «

That is, the hard constraint tightness is the fraction of tuples that are forbidden by the constraint with respect to the total number of tuples. For the generated instances under consideration here, for a particular problem, all hard constraints have the same tightness, therefore in this context ht denotes the tightness of all the hard constraints in the problem. The work in [GMP+01] highlights some problems with

generation procedures for random binary (hard) constraint problems, for which our generation procedure conforms to model C in [GMP+01], however the primary focus of the experimentation is on the soft constraint aspect of the generated problems.

Given the values for these parameters for hard constraints, and given that each variable domain is of size m, we can calculate the *expected number of consistent solutions* [SD96] for any given problem, which is defined as follows

Definition 5.26 » Expected number of consistent solutions (es)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *expected number of consistent solutions* is given by

— $es = ss \times ((1-ht)^{hc})$ «

We now look at some properties of the soft constraints of a problem. Firstly, we define the *soft constraint count*.

Definition 5.27 » Soft Constraint Count (sc)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *soft constraint count* is given by

— $sc = |\mathcal{C}_S|$ «

We also look at the size of the scale from which the preference values are chosen.

Definition 5.28 » Size of scale T
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, let z denote the maximum value of the preference scale T, so that the scale T is given by

— $T = (1, \ldots, z)$ «

We may also wish to express the number of soft constraints as a parameter of the problem size. Therefore we have the soft constraint density as follows.

Definition 5.29 » Soft Constraint Density (sd)
For a problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, the *soft constraint density* is given by

— $sd = \dfrac{sc}{(n * (n-1)/2)}$ «

That is, the soft constraint density is the fraction of soft constraints in the problem with respect to the maximum number of constraints in a normalised problem with

binary soft constraints.

Generation Parameters

We now define the parameters that are passed to the constraint generator to generate the random problems. To generate a problem containing hard constraints, i.e., such that $\mathcal{C}_H \neq \emptyset$, either the hc, hd or es parameter needs to be specified. To generate a problem containing soft constraints, i.e., such that $\mathcal{C}_S \neq \emptyset$, the sc parameter needs to be specified. If neither hard nor soft constraints are specified, then we have a trivial problem with no constraints.

n specifies the number of variables in the problem.

m specifies the size of variable domains.

hc specifies the number of hard constraints in the problem.

hd specifies the hard constraint density, where $hd \in [0, 1]$.

ht specifies the tightness of each hard constraint, where $ht \in [0, 1]$.

es specifies the expected number of solutions

sc specifies the number of soft constraints.

sd specifies the soft constraint density, where $sd \in [0, 1]$.

k specifies the preference value of non-satisfaction, i.e., if a tuple is associated this value by a constraint, then the tuple does not satisfy the constraint.

z specifies the maximum value of scale T.

5.7 Experimental Results

In this section, we look at some experimental results for solving some general constraints problems using the Sorted-Pareto dominance relation. First we perform a brief comparison of the resulting Pareto and Sorted-Pareto optimal sets for some random instances of different sizes. We perform an evaluation of the six algorithms discussed, using some randomly generated problem instances and some modified benchmark instances. For the random instances, we investigate the effect of varying some generation parameters, such as the size of the problems, the size of the preference scale used, and the number of soft constraints. Finally we compare the

sizes of the resulting Sorted-Pareto optimal sets against the sizes of the resulting Lexicographic-Max optimal sets for different problem instances.

5.7.1 Experimental Setup

The experiments in this section were run on a node on the 4C cluster [Cen13], which has a dual Intel Xeon E5430 Processor (2.66Ghz) machine, with 12GB RAM. Table 5.1 shows the relevant problem generation parameters from Section 5.6.2 for all the problem instance sets generated.

Table 5.1: Table of parameters used for generating the sets of problem instances for the experimental results.

Name	size	n	d	hd	ht	z	sc	sd
Set A	50	$10, 12, \ldots, 20$	2	0.06	0.25	10	-	0.25
Set A - II	50	$24, 28, \ldots, 40$	2	0.06	0.25	10	-	0.25
Set B	50	$20, 24, \ldots, 40$	2	0.06	0.25	3	25	-
Set C	50	25	2	0.06	0.25	3	$5, 10, \ldots, 30$	-
Set D	50	25	2	0.06	0.25	$3, 4, \ldots, 8$	10	-

5.7.2 Comparing Pareto and Sorted-Pareto optimal sets

In this section, we compare the sizes of the resulting Pareto optimal and Sorted-Pareto optimal sets for some general constraint problem instances. Table 5.2 shows the average number of consistent solutions $\langle \text{Sol} \rangle$, the average number of Pareto non-dominated solutions $\langle \text{O}_\text{P} \rangle$, and the average number of Sorted-Pareto non-dominated solutions $\langle \text{O}_\text{SP} \rangle$, for the instances in Set A (see Table 5.1), where we vary n, i.e., the size of the problems, and a set of 50 problems was generated for each n. For these parameters, the number of consistent solutions grows exponentially with the size of the problem, given the range of the values of n used, (however at bigger values of n, the rate of growth of the number of consistent solutions will drop off and eventually decline). The number of soft constraints grows as a parameter of the problem size n, as a result of specifying the sd parameter, and the maximum scale value is $z = 10$. For these small size problems we can see that for increasing values of n, the size of the set of Pareto non-dominated solutions grows very rapidly, whereas the size of the set of Sorted-Pareto non-dominated solutions grows much slower.

Table 5.3 shows the average number of consistent solutions $\langle \text{Sol} \rangle$ and the average number of Sorted-Pareto non-dominated solutions $\langle \text{O}_\text{SP} \rangle$ for the instances in Set

Table 5.2: The average number of solutions $\langle \text{Sol} \rangle$, the average number of Pareto non-dominated solutions $\langle O_P \rangle$, and the average number of Sorted-Pareto non-dominated solutions $\langle O_{SP} \rangle$, for the instances in Set A, varying the size of the problems $n = 10, 12, 14, 16, 18, 20$.

Set A	$n = 10$	$n = 12$	$n = 14$	$n = 16$	$n = 18$	$n = 20$
$\langle \text{Sol} \rangle$	432	1,292	3,830	8,785	19,974	47,018
$\langle O_P \rangle$	38	141	419	1,120	3,838	13,443
$\langle O_{SP} \rangle$	7	13	22	28	43	75

A - II, which are over larger values of n. We can see that for these larger instances the sets O_{SP} are still moderately sized.

Table 5.3: The average number of consistent solutions $\langle \text{Sol} \rangle$ and average number of Sorted-Pareto non-dominated solutions $\langle O_{SP} \rangle$, for the instances in Set A, for larger problem sizes $n = 24, 28, 32, 36, 40$.

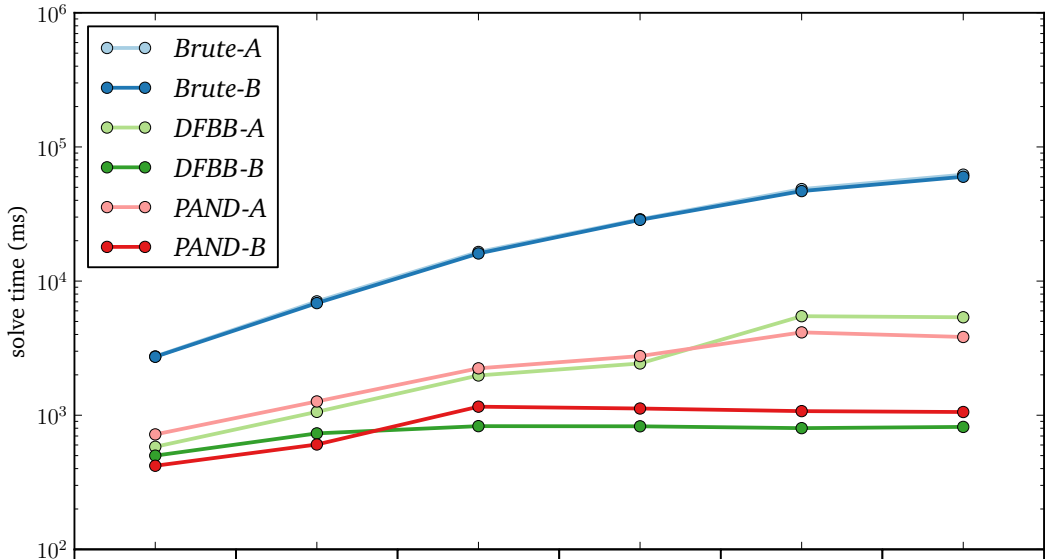
Set A	$n = 24$	$n = 28$	$n = 32$	$n = 36$	$n = 40$
$\langle \text{Sol} \rangle$	126,500	417,507	702,907	1,476,080	1,759,033
$\langle O_{SP} \rangle$	128	216	247	352	403

5.7.3 Comparing Sorted-Pareto algorithms

In this section, we look at some experimental results for solving Sorted-Pareto constraints problems. We compare the six different algorithms that were described in Section 5.4, by generating some random problems and evaluating the performances of the algorithms where we vary different problem parameters such as the size of the problems and the number of soft constraints. We also look at the sizes of the resulting Sorted-Pareto optimal sets and the number of Sorted-Pareto equivalence classes for these problems.

Varying problem size

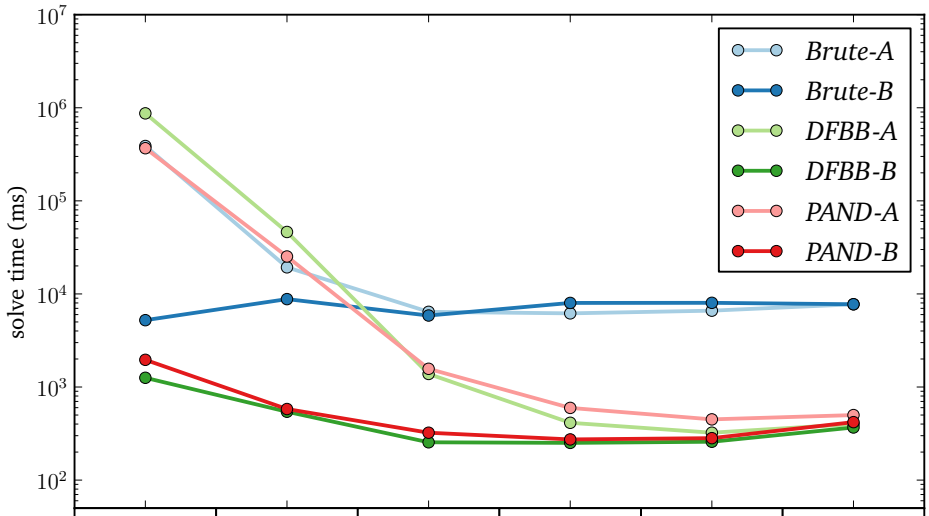
Figure 5.4 shows the average solve times for the algorithms for the instances in Set B (see Table 5.1). In these instances, the size of the problems is varied, i.e., a set of 50 problems was generated for each of $n = 20, 24, 28, 32, 36, 40$. The table accompanying Figure 5.4 shows the average number of solutions $\langle \text{Sol} \rangle$, the average number of Sorted-Pareto optimal solutions $\langle O_{SP} \rangle$, the average number of Sorted-Pareto optimal equivalence classes $\langle O_{[SP]} \rangle$, and the ratio between the number of optimal solutions and optimal equivalence classes.



	n=20	n=24	n=28	n=32	n=36	n=40
$\langle$Sol$\rangle$	44260.48	125977.84	356328.00	764423.20	1218371.52	1462344.32
$\langle O_{SP} \rangle$	50.90	89.12	141.18	154.26	316.20	335.18
$\langle O_{[SP]} \rangle$	18.66	21.32	16.62	13.90	11.22	9.58
Ratio	2.73	4.18	8.49	11.10	28.18	34.99

Figure 5.4: Graph of average solve times for algorithms *Brute-A*, *Brute-B*, *DFBB-A*, *DFBB-B*, *PAND-A*, and *PAND-B*, for the instances in Set B, varying the size of the problems $n = 20, 24, 28, 32, 36, 40$. The table shows the average number of solutions $\langle$Sol$\rangle$, average number of Sorted-Pareto optimal solutions $\langle O_{SP} \rangle$, average number of Sorted-Pareto optimal equivalence classes $\langle O_{[SP]} \rangle$, and the ratio between $\langle O_{SP} \rangle$ and $\langle O_{[SP]} \rangle$.

As we can see in Figure 5.4, both the set based and equivalence handling brute force algorithms, *Brute-A* and *Brute-B*, perform similarly (the graphed lines of solve times for the two algorithms are almost identical), and the solve times grow with the size of the problems. The solve times of the set-based algorithms *DFBB-A* and *PAND-A* also appear to grow with the size of the problems, however the solve times for the equivalence-handling algorithms *DFBB-B* and *PAND-B* appear to be constant even as the size of the problems increase. The table attached to Figure 5.4 may indicate a reasonable explanation for this, as for these particular problem instances, even though the average number of Sorted-Pareto non-dominated solutions is growing as the sizes of the problems grow, the number of equivalence classes is decreasing, therefore there is a smaller number of distinct preference values for the algorithm to handle during the search when performing dominance checks.



	sc=5	sc=10	sc=15	sc=20	sc=25	sc=30
$\langle$Sol$\rangle$	188787.52	188787.52	188787.52	188787.52	188787.52	188787.52
$\langle O_{SP} \rangle$	8470.96	1084.80	214.08	84.24	53.90	24.10
$\langle O_{[SP]} \rangle$	1.10	1.32	1.56	1.84	2.24	2.30
Ratio	7700.87	821.82	137.23	45.78	24.06	10.48

Figure 5.5: Graph of average solve times for algorithms *Brute-A*, *Brute-B*, *DFBB-A*, *DFBB-B*, *PAND-A*, and *PAND-B*, for the instances in Set C, varying the number of soft constraints $sc = 5, 10, 15, 20, 25$. The table shows the average number of solutions $\langle$Sol$\rangle$, average number of Sorted-Pareto optimal solutions $\langle O_{SP} \rangle$, average number of Sorted-Pareto optimal equivalence classes $\langle O_{[SP]} \rangle$, and the ratio between $\langle O_{SP} \rangle$ and $\langle O_{[SP]} \rangle$.

Varying number of soft constraints

Figure 5.5 shows the average solve times for the algorithms for the instances in Set C (see Table 5.1). In these instances, the size of the problems is constant, but the number of soft constraints is varied. Firstly, a set of 50 problems was generated which contained only hard constraints, and from this set, to vary the number of soft constraints, a new set of problems was generated for each of $sc = 5, 10, 15, 20, 25$. The table shown in Figure 5.5 shows the average number of solutions $\langle$Sol$\rangle$, the average number of Sorted-Pareto optimal solutions $\langle O_{SP} \rangle$, the average number of Sorted-Pareto optimal equivalence classes $\langle O_{[SP]} \rangle$, and the ratio between the number of optimal solutions and optimal equivalence classes.

In these instances, the set-based algorithms *DFBB-A* and *PAND-A* are outperformed by the equivalence-handling algorithms *DFBB-B* and *PAND-B*. For the instances where there is a higher average number of Sorted-Pareto non-dominated solutions,

for example when $sc = 10$ we have $\langle O_{SP} \rangle = 1084$, we can see that the equivalence handling algorithms perform much better than the set based algorithms. We also observe that, as the number of soft constraints increases, $\langle O_{SP} \rangle$ decreases, and the average solve times of the set based algorithms *DFBB-A* and *PAND-A* approach the solve times of the equivalence-handling algorithms *DFBB-B* and *PAND-B*.

5.7.4 Non-random problems

In this section, we compare the performance of the algorithms for some modified benchmark instances from the CELAR Radio-Link Frequency Assignment problem benchmark (RLFAP problem) [CdGL+99, Scha, Schb]. The instances used are shown in Table 5.4, and have been modified by adding some hard constraints to limit the number of solutions to around 100,000.

The table shows for each problem: the number of variables n, the maximum domain sizes d, the size of the scale $|T|$, the number of soft constraints sc, the number of solutions |Sol|, the number of Sorted-Pareto optimal solutions $|O_{SP}|$, the number of Sorted-Pareto optimal equivalence classes $|O_{[SP]}|$, and the ratio between the number of optimal solutions and optimal equivalence classes.

Table 5.4: The number of Sorted-Pareto optimal solutions $|O_{SP}|$ and the number of Sorted-Pareto optimal equivalence classes $|O_{[SP]}|$ for the modified instances from the CELAR Radio-Link Frequency Assignment problem benchmark, where hard constraints have been added to limit the number of solutions to around 100,000.

Name	n	d	$\|T\|$	sc	|Sol|	$\|O_{SP}\|$	$\|O_{[SP]}\|$	Ratio
CELAR6-SUB0*	16	44	5	207	101660	17	9	1.89
CELAR6-SUB1*	14	44	5	300	91562	35	32	1.09
CELAR6-SUB2*	16	44	5	353	100783	20	20	1.00
CELAR6-SUB3*	18	44	5	421	96611	23	19	1.21
CELAR6-SUB4*	22	44	5	477	91994	36	27	1.33
CELAR7-SUB0*	16	44	5	188	91010	11	9	1.22
CELAR7-SUB1*	14	44	5	300	97437	19	17	1.12
CELAR7-SUB2*	16	44	5	353	93569	31	29	1.07
CELAR7-SUB3*	18	44	5	421	101851	42	34	1.24
CELAR7-SUB4*	22	44	5	477	97185	26	23	1.13

Figure 5.6 shows the solve time of the algorithms for the instances given in Table 5.4. In these instances the set based algorithms perform very similarly to their equivalence handling counterparts. This could be explained by the fact that the ratio of optimal solutions to optimal equivalence classes is low for each instance, indicating that there is very little gain from the equivalence handling approach. The depth-first branch and bound algorithms *DFBB-A* and *DFBB-B*, which use only a

lower bound, perform the best, and the *PAND-A* and *PAND-B* algorithms which use both a lower and upper bound, perform the worst.

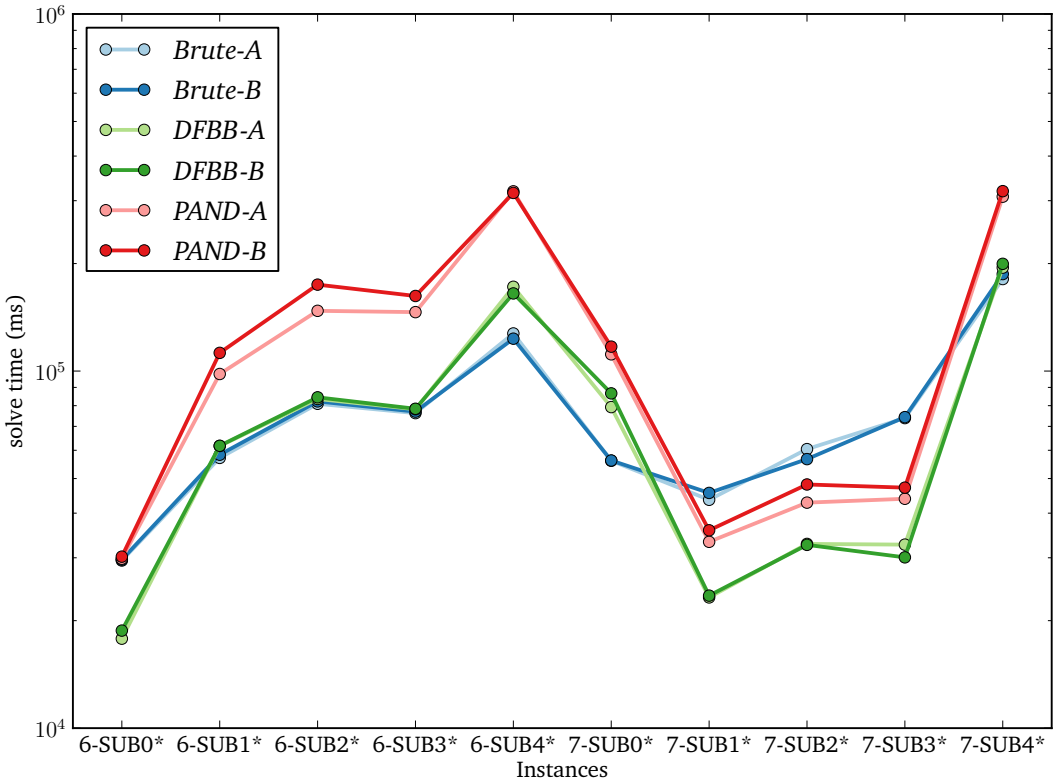


Figure 5.6: Solve times for algorithms *Brute-A*, *Brute-B*, *DFBB-A*, *DFBB-B*, *PAND-A*, and *PAND-B*, for the modified CELAR RLFAP instances given in Table 5.4.

5.7.5 Discussion of the algorithms

The Brute force algorithms *Brute-A* and *Brute-B* are a naive approach, and Figure 5.4 shows that these algorithms perform relative to the size of the problems. However as we saw in Figure 5.6, for certain instances these algorithms can perform similarly to a branch and bound approach.

The results in Figure 5.4 and Figure 5.5 show that the equivalence handling versions of the branch and bound algorithms *DFBB-B* and *PAND-B* perform better than their set based versions *DFBB-A* and *PAND-A*. However the measure of improvement depends on the size of the Sorted-Pareto optimal set and the number of Sorted-Pareto equivalence classes, and the ratio between these numbers. For higher ratios and smaller numbers of Sorted-Pareto equivalence classes, the equivalence handling algorithms are much better since they are required to perform a smaller number

of dominance checks than the set based algorithms. As this ratio decreases then the number of dominance checks performed by both the equivalence handling and set based algorithms converge, and as we observe in Figure 5.5 that the overall performances of the algorithms converge.

Overall the results show that equivalence handling branch and bound algorithm *DFBB-B* performed better than the other algorithms. The *PAND-A* and *PAND-B* branch and bound algorithms, which uses both a lower and upper bound, generally did not perform better than the lower bound only versions *DFBB-A* and *DFBB-B*. In Figure 5.4 for $n = 36$ and $n = 40$ we observe that for the set based algorithms, *PAND-A* does perform marginally better than *DFBB-A*, indicating that there are instances where the upper bound is useful, but in these instances the set based algorithms are bettered anyway by the equivalence handling algorithms.

5.7.6 Varying Size of Preference Scale

As we saw for the instances in Figure 5.5, varying the size of the preference vector has an effect on the number of resulting Sorted-Pareto optimal solutions and Sorted-Pareto optimal equivalence classes. In these instances, for a scale z of size 3, as the size of the preference vector increases, the average number of Sorted-Pareto equivalence classes also increases, indicating more variance in the preference levels of the Sorted-Pareto optimal solutions. This also results in a decrease in the average number of Sorted-Pareto optimal solutions, possibly because of this variance resulting in more decisions that are dominated.

We now look at the effect of varying the scale size, i.e., the size of the scale T (which is given by the z parameter) on some generated problems. Table 5.5 shows the average number of solutions $\langle \text{Sol} \rangle$, the average number of Sorted-Pareto optimal solutions $\langle \text{O}_{\text{SP}} \rangle$, the average number of Sorted-Pareto optimal equivalence classes $\langle \text{O}_{[\text{SP}]} \rangle$, and the ratio between the number of optimal solutions and optimal equivalence classes, for the instances in Set D (see Table 5.1). In these instances, both the size of the problems and the size of the preference vectors are constant, but the size of the preference scale T is varied. Firstly, a set of 50 problems was generated for a scale of size 8 (denoted by $z = 8$) and from this set, the scale used in these problems was rescaled to generate a new set of problems for each of $z = 3, 4, 5, 6, 7$.

In these instances, as the size of the scale T increases, the average number of Sorted-Pareto equivalence classes also increases, and we also observe a decrease in the average number of Sorted-Pareto optimal solutions. Similar to the results in Figure

Table 5.5: The average number of solutions $\langle \text{Sol} \rangle$, the average number of Sorted-Pareto optimal solutions $\langle \text{O}_{\text{SP}} \rangle$, the average number of Sorted-Pareto optimal equivalence classes $\langle \text{O}_{[\text{SP}]} \rangle$, and the ratio between the number of optimal solutions and optimal equivalence classes, for the instances in Set D, where the size of the scale T is varied (by parameter z).

Set D	$z=3$	$z=4$	$z=5$	$z=6$	$z=7$	$z=8$
$\langle \text{Sol} \rangle$	188787.52	188787.52	188787.52	188787.52	188787.52	188787.52
$\langle \text{O}_{\text{SP}} \rangle$	939.00	647.76	563.44	570.96	481.76	383.56
$\langle \text{O}_{[\text{SP}]} \rangle$	1.28	1.42	1.80	2.00	2.18	2.22
Ratio	733.59	456.17	313.02	285.48	220.99	172.77

5.5 where the size of the preference vector is varied, we can see that increasing the size of the scale T results in more Sorted-Pareto optimal equivalence classes and less Sorted-Pareto optimal solutions.

5.7.7 Comparing Sorted-Pareto and Lexicographic-Max Ordering

In this section, we compare Sorted-Pareto dominance with the Lexicographic-Max ordering as given in Section 2.6. For a general constraints problem $P = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_H, \mathcal{C}_S, \mathcal{P} \rangle$, where the preference degree structure $\mathcal{P} = \langle I, \otimes, \preceq \rangle$ is defined with the Lexicographic-Max ordering $\preceq_{\text{LMX}}$ (see Definition 2.29), then the resulting order on $\text{Sol}(P)$ is a total preorder.

We look at two tables of results for the Lexicographic-Max ordering. Table 5.6 shows the average number of Lexicographic-Max optimal solutions $\langle \text{O}_{\text{LMX}} \rangle$ for the instances in Set B (originally from Figure 5.4). Table 5.7 shows the number of Lexicographic-Max optimal solutions $\langle \text{O}_{\text{LMX}} \rangle$ for the modified CELAR RLFAP instances (originally given in Table 5.4).

Table 5.6: Comparing the average number of Sorted-Pareto optimal solutions $\langle \text{O}_{\text{SP}} \rangle$ and the average number of Lexicographic-Max optimal solutions $\langle \text{O}_{\text{LMX}} \rangle$ for the instances in Set B.

Set B	$n=20$	$n=24$	$n=28$	$n=32$	$n=36$	$n=40$
$\langle \text{Sol} \rangle$	44260.48	125977.84	356328	764423.20	1218371.52	1462344.32
$\langle \text{O}_{\text{SP}} \rangle$	50.90	89.12	141.18	154.26	316.20	335.18
$\langle \text{O}_{[\text{SP}]} \rangle$	18.66	21.32	16.62	13.90	11.22	9.58
$\langle \text{O}_{\text{LMX}} \rangle$	2.60	4.16	7.12	9.82	30.46	29.62

In both set of results, the number of Lexicographic-Max optimal solutions is much smaller than the number of Sorted-Pareto optimal solutions. As mentioned in the

Table 5.7: Comparing the number of Sorted-Pareto optimal solutions $|O_{SP}|$ and the number of Lexicographic-Max optimal solutions $|O_{LMX}|$ for the modified CELAR instances.

Name	n	d	$\|T\|$	sc	|Sol|	$\|O_{SP}\|$	$\|O_{[SP]}\|$	$\|O_{LMX}\|$
CELAR6-SUB0*	16	44	5	207	101660	17	9	1
CELAR6-SUB1*	14	44	5	300	91562	35	32	1
CELAR6-SUB2*	16	44	5	353	100783	20	20	1
CELAR6-SUB3*	18	44	5	421	96611	23	19	1
CELAR6-SUB4*	22	44	5	477	91994	36	27	1
CELAR7-SUB0*	16	44	5	188	91010	11	9	2
CELAR7-SUB1*	14	44	5	300	97437	19	17	2
CELAR7-SUB2*	16	44	5	353	93569	31	29	1
CELAR7-SUB3*	18	44	5	421	101851	42	34	3
CELAR7-SUB4*	22	44	5	477	97185	26	23	1

introduction to the thesis in Section 1.1, we consider situations where the task is to support a decision maker by presenting a subset of decisions from a larger set of initial decisions. Since Lexicographic-Max is a total preorder, then choosing to present the Lexicographic-Max optimal decisions would result in only presenting one or a set of equivalent decisions to a decision maker. Also, as shown in Section 2.6.2, the Lexicographic-Max relation does not consider all preference levels associated with a decision, it ignores some preference values. The number of Sorted-Pareto optimal equivalence classes in the results indicate the number of distinct preference levels of the solutions in the set of Sorted-Pareto optimal decisions, so we can present a more varied set of decisions to a decision maker by showing all the Sorted-Pareto optimal decisions or just a representative from each equivalence class.

5.8 An Extension to Sorted-Pareto Dominance

In this section, we look at an extension to the Sorted-Pareto relation, which looks at further refining a set of Sorted-Pareto non-dominated decisions. When we have a set of Sorted-Pareto non-dominated solutions O_{SP} to a particular problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, we can further order this set of optimal elements using the MinMax-Sorted-Pareto relation $\prec_{SP}^{max}$, which is defined as follows.

Definition 5.30 » MinMax Sorted-Pareto preferred
For all $\alpha, \beta \in O_{SP}$, $\alpha \prec_{SP}^{max} \beta$, if and only if

— $\max(\rho(\alpha)) < \max(\rho(\beta))$ «

From this, we can define MinMax Sorted-Pareto optimal as follows.

Definition 5.31 » MinMax Sorted-Pareto preferred
A decision $\alpha \in O_{SP}$ is *MinMax Sorted-Pareto optimal* if and only if

— for all $\beta \in O_{SP}$, $\alpha \prec_{SP}^{max} \beta$. «

The set of MinMax Sorted-Pareto optimal decisions is denoted by O_{SP}^{max}.

Let us look at an example.

Example 5.6 ► MinMax Sorted-Pareto example**.**
Consider a multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$ *where:*

- $\mathcal{A} = \{\alpha, \beta\}$*, i.e., there are two decisions,*
- $\mathcal{S} = \{1, 2, 3\}$*, i.e., there are three preference values to consider*
- $T = \{low, med, hi\}$*, ordered by* $\leq$*, where again we wish to minimise.*

Suppose that the resulting preference levels are:

- $\rho(\alpha) = \{med, med, med\}$
- $\rho(\beta) = \{hi, low, hi\}$

We can see that neither decision Sorted-Pareto dominates the other, so we have that

- $O_{SP} = \{\alpha, \beta\}$.

Now, supposing we use MinMax Sorted-Pareto dominance, we can see that

- $\alpha \prec_{SP}^{max} \beta$*, since* $\max(\alpha) < \max(\beta)$.

Therefore we have that the optimal set with respect to Minmax Sorted-Pareto is given by

- $O_{SP}^{max} = \{\alpha\}$. ▲

This relation forms a total preorder on O_{SP}, and could be used as a tiebreaker between decisions, to present a single solution or very small set of solutions to a decision maker. It could also be considered as an *egalitarian* approach, since it prefers the decisions that minimise the maximum negative preferences, for example, if the preferences represent some costs which are to be minimised, then the relation $\prec_{SP}^{max}$ prefers decisions with smaller maximum costs.

MinMin-Sorted-Pareto

Briefly, an alternative approach for refining the Sorted-Pareto optimal set in a similar way could be to minimise the *minimum* negative preferences (similar to a Maximax relation [BT97] for positive preferences). Let us call this the *MinMin-Sorted-Pareto* relation, where the Sorted-Pareto optimal decisions with smaller minimium preference values are preferred. For example, in decision making under uncertainty where the aspects represent possible scenarios that could occur, the MinMin-Sorted-Pareto optimal decisions would be the ones that could result in the lowest cost.

5.8.1 Experimental Results

In this section we look at some experimental results for generating the MinMax Sorted-Pareto optimal sets for some problem instances. We look at two sets of results. Table 5.8 shows again the experimental results for the instances in Set B from Table 5.6, in addition we show the average number of MinMax-Sorted-Pareto optimal solutions $\langle O_{\underset{SP}{max}} \rangle$, and the resulting number of Sorted-Pareto equivalence classes for these solution, denoted by $\langle O_{[\underset{SP}{max}]} \rangle$. Table 5.9 shows again the experimental results for the modified CELAR RLFAP instances from Table 5.7, and in addition we show the number of MinMax-Sorted-Pareto optimal solutions $|O_{\underset{SP}{max}}|$ and the resulting number of Sorted-Pareto equivalence classes $|O_{[\underset{SP}{max}]}|$ for each instance.

Table 5.8: Comparing the average number of Sorted-Pareto optimal solutions $\langle O_{SP} \rangle$, and the average number of Sorted-Pareto MinMax optimal solutions $\langle O_{\underset{SP}{max}} \rangle$ for the instances in Set B.

Set B	$n = 20$	$n = 24$	$n = 28$	$n = 32$	$n = 36$	$n = 40$
$\langle Sol \rangle$	44260.48	125977.84	356328	764423.20	1218371.52	1462344.32
$\langle O_{SP} \rangle$	50.90	89.12	141.18	154.26	316.20	335.18
$\langle O_{[SP]} \rangle$	18.66	21.32	16.62	13.90	11.22	9.58
$\langle O_{LMX} \rangle$	2.60	4.16	7.12	9.82	30.46	29.62
$\langle O_{\underset{SP}{max}} \rangle$	14.78	25.08	48.04	51.14	176.28	130.18
$\langle O_{[\underset{SP}{max}]} \rangle$	5.74	6.66	5.46	4.54	5.64	3.82

We can see that in Table 5.8, the average number of MinMax Sorted-Pareto optimal solutions $\langle O_{\underset{SP}{max}} \rangle$ is significantly smaller than the number of Sorted-Pareto optimal solutions. However in Table 5.9, there is only one instance (CELAR7-SUB0*) where there is a reduction in the optimal set size using MinMax Sorted-Pareto. As mentioned in 5.7.7, where we consider situations where we are trying to narrow down

Table 5.9: Comparing the number of Sorted-Pareto optimal solutions $|O_{SP}|$, and the number of Sorted-Pareto MinMax optimal solutions $|O^{max}_{SP}|$ for the modified CELAR instances.

Name	n	d	$\|T\|$	sc	|Sol|	$\|O_{SP}\|$	$\|O_{[SP]}\|$	$\|O_{LMX}\|$	$\|O^{max}_{SP}\|$	$\|O_{\lfloor^{max}_{SP}\rfloor}\|$
CELAR6-SUB0*	16	44	5	207	101660	17	9	1	17	9
CELAR6-SUB1*	14	44	5	300	91562	35	32	1	35	32
CELAR6-SUB2*	16	44	5	353	100783	20	20	1	20	20
CELAR6-SUB3*	18	44	5	421	96611	23	19	1	23	19
CELAR6-SUB4*	22	44	5	477	91994	36	27	1	36	27
CELAR7-SUB0*	16	44	5	188	91010	11	9	2	6	4
CELAR7-SUB1*	14	44	5	300	97437	19	17	2	19	17
CELAR7-SUB2*	16	44	5	353	93569	31	29	1	31	29
CELAR7-SUB3*	18	44	5	421	101851	42	34	3	42	34
CELAR7-SUB4*	22	44	5	477	97185	26	23	1	26	23

the optimal set to present to a decision maker, we could present the set of MinMax-Sorted-Pareto optimal decisions or alternatively, a set of representatives from each Minimax-Sorted-Pareto optimal equivalence class.

5.9 Related Works

In Section 2.6, we look at the connections betweeen Sorted-Pareto dominance and other preference relations; here we look at some more related works in regards to algorithms and methods for approximating the Pareto optimal set and solving other partially ordered problems. The notion of Sorted-Pareto appears in [LM95], which focuses on the elicitation of the preferences and the normalisation of different criteria scales, whereas in this thesis we assume that such a normalisation process has occurred. It is also called *Ordered Pareto* [KP08] or *Symmetric Pareto* [DPT13], and is used for handling preferences and comparing alternatives using possibilistic logic. As we show in Section 5.2.1, Sorted-Pareto is related to preference based search for generating sets of optimal solutions for shortest path problems [PS05, BS06]. It also appears in social welfare theory [TSS09] where it is applied to ordered income distributions.

As we show in Proposition 2.4, the Sorted Pareto relation extends the Pareto dominance relation, and computing the Sorted-Pareto optimal set is viable when preference level scales are commensurate, since calculating the Pareto optimal set can be prohibitive as we experienced in our results in Section 5.7.2. Some other work that approximates the Pareto optimal set in constraints problems includes approaches that utilise quantitative information to perform a sum of weights on the preference

vectors [TF02], and other approximating branch and bound algorithms [Gav02] similar to what is used in Section 5.4. Other different approaches for approximating a set of Pareto optimal solutions include: approaches that use *AND/OR* search spaces in branch and bound algorithms [Mar11], approaches that use multi-objective influence diagrams [MRW12], and approaches that incorporate additional trade-offs to reduce the size of the optimal set [MRW13].

Other algorithms for handling partially ordered soft constraints include branch and bound algorithms for partially-ordered constraint optimisation problems (PCOP) [Gav01] and a branch and bound algorithm for partially ordered degrees of preferences [WF08]. For the Leximin preference relation, there are branch and bound algorithms for the computation of Leximin optimal solutions in Constraint Networks [BL09].

5.10 Chapter Conclusion

In this chapter, we defined an extension to Pareto dominance called Sorted-Pareto dominance. We gave a semantics for the relation, showing it is very relevant to a situation where we have a weighted constraints problem [RvBW06, Ch. 9] (or, similarly, a GAI decomposition [BG95]) but the numerical values are only on an ordinal scale. Theorems 5.1 and 5.2 show that a decision Sorted-Pareto dominates another if and only if it dominates the other in all compatible standard weighted constraints problems.

We also explored Sorted-Pareto in the context of Soft Constraints, and we gave three different depth-first algorithms, along with some variations of these algorithms, for providing a set of Sorted-Pareto optimal solutions to general constraints problems which involve both hard and soft constraints. The experimental results showed that often the resulting set of optimal solutions is relatively small, and of certainly a much more manageable size than the set of Pareto optimal solutions. For our algorithm implementations, the algorithm with lower bound pruning generates an order of magnitude speedup in some instances. For the algorithm which uses an upper bound, a similar approach was shown to be useful in an application with comparative preferences [WT11], however we found that in these particular soft constraint problem instances, the algorithm did not improve solve times.

In our experimental results, we found that the number of soft constraints and also the size of the preference scale used in the problem instances was a factor

in the resulting number of Sorted-Pareto equivalence classes and the number of Sorted-Pareto optimal solutions. We also compared the sizes of the resulting optimal sets to the sizes of the resulting Lexicographic-Max optimal set. We argue, that in some settings, it may be better to present a more varied set of Sorted-Pareto optimal solutions rather than a single Lexicographical-Max solution that, as shown in Section 2.6.2, can ignore some of the preferences associated to solutions, whereas the Sorted-Pareto optimal solutions consider all the preference values associated to solutions.

Chapter 6

Sorted-Pareto Dominance and Qualitative Notions Of Optimality

6.1 Introduction

In a decision-making task, it is often the case that the basis for comparing decisions involves more than one preference value (e.g., evaluations of multiple criteria in multi-criteria decision making, evaluations by more than one agent in multi-agent decision making, or considerations of different states in decision making under uncertainty), and therefore we have a preference vector for each decision. If the scale T is quantitative, or we have information that gives a quantitative mapping for T, e.g., we have a mapping $f : T \to \boldsymbol{R}^+$, then the decisions could be compared by summing the preference vector values and seeing which decisions have the smallest sum of costs or the largest sum of utilities, as seen in Section 2.6.5.

However, often the preference information available is only of an ordinal or qualitative nature, as it can be easier to obtain such information, e.g., there may be uncertainty over exact values, or it may be easier to elicit qualitative preference information from a decision maker [MMO02]. As seen in Chapter 5, the Sorted-Pareto preference relation relies only on ordinal or qualitative information, and therefore can be used in these qualitative decision making situations. In addition, for any mapping $f : T \to \boldsymbol{R}^+$, where f is *strictly monotonic* with respect to the scale T, i.e., for all $u, v \in T$, $u \leq v \Leftrightarrow f(u) \leq f(v)$, the Sorted-Pareto relation gives an ordering that is compatible with any such function f.

As we saw in Chapter 4, in a partially ordered setting, such as in the situation just described, there can be different natural notions of optimality. The framework in Chapter 4 describes some of these notions, for qualitative decision making under uncertainty, where there are different possible scenarios in a given problem. This gives us classes of decisions that are not dominated by any other decision, decisions that are possibly optimal or possibly strictly optimal, (i.e., optimal in some scenario), and decisions that are optimal in all scenarios. As shown in Section 5.3, Sorted-Pareto connects to Weighted Constraints Satisfaction Problems (WCSP) [RvBW06, Ch. 9] and Bayesian Networks [Pea88] where we only have ordinal information, and in these frameworks the possibly optimal decisions are those that are min-sum optimal for some compatible WCSP, or are the complete assignments that are most probable in some compatible Bayesian Network.

In this chapter, we look at the relationship between the Sorted-Pareto preference relation and Min-sum of weights orderings, and we examine this relationship as an instance of the MODS framework from Chapter 4. The chapter outline is as follows. In Section 6.2 we provide some preliminaries for the chapter, revisiting

the Sorted-Pareto preference relation and the min-sum of weights ordering, and the relationship between the two orderings as given in Section 5.3. In Section 6.3 we use the MODS framework from Chapter 4 to describe this relationship between Sorted-Pareto dominance and min-sum of weights, and we examine the different natural notions of optimality and the optimality classes that occur in the Sorted-Pareto case. In Section 6.4, we show how to generate these optimality classes for the Sorted-Pareto case, and in Section 6.5 we present some experimental results which look at the resulting sets computed for the different notions of optimality.

6.2 Preliminaries

In this section, we recall in brief the relevant definitions and setup from the prequel to give the setting for the work of this chapter. We assume a multi-aspect decision problem $P = \langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$ as given in Definition 2.11, where for each decision $\alpha \in \mathcal{A}$ we have a preference vector $v(\alpha) = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ of m preference values for α. The preference vector $v(\alpha)$ of decision α, sorted in non-descending order, is denoted by $v(\alpha)^{\uparrow}$.

For Sorted-Pareto dominance, we recall from Definitions 2.19 to 2.21 that, for all $\alpha, \beta \in \mathcal{A}$, $\alpha \preccurlyeq_{\text{SP}} \beta$ if and only if $v(\alpha)^{\uparrow} \leq v(\beta)^{\uparrow}$, $\alpha \prec_{\text{SP}} \beta$ if and only if $v(\alpha)^{\uparrow} < v(\beta)^{\uparrow}$, and $\alpha \equiv_{\text{SP}} \beta$ if and only if $v(\alpha)^{\uparrow} = v(\beta)^{\uparrow}$, where $\leq$, $<$ and $=$ are defined over preference vectors, as given in Definition 2.14. A decision $\alpha \in \mathcal{A}$ is Sorted-Pareto optimal if and only if there is no $\beta \in \mathcal{A}$ such that $\beta \prec_{\text{SP}} \alpha$.

For quantitative scales, or for when there is a function $f : T \to \mathbb{R}^+$ mapping a qualitative scale to a quantitative one, we recall here from Section 2.6.5 the definitions for Min-sum preferred, strictly Min-sum preferred, and Min-sum equivalent. For all $\alpha, \beta \in \mathcal{A}$, for some function $f : T \to \mathbb{R}^+$, α is Min-sum preferred (with respect to f) to β, written as $\alpha \leq_f \beta$, if and only if $\sum_{i=1}^{m} f(\alpha_i) \leq \sum_{i=1}^{m} f(\beta_i)$. We have $\alpha <_f \beta$ if and only if $\sum_{i=1}^{m} f(\alpha_i) < \sum_{i=1}^{m} f(\beta_i)$, and we have $\alpha =_f \beta$ and only if $\sum_{i=1}^{m} f(\alpha_i) = \sum_{i=1}^{m} f(\beta_i)$. For some function $f : T \to \mathbb{R}^+$, a decision $\alpha \in \mathcal{A}$ is *Min-sum optimal* (with respect to f) if and only if $\alpha \leq_f \beta$ for all $\beta \in \mathcal{A}$.

Finally, in brief, from Section 4.2, a multiple-ordering decision structure $\mathcal{G} = \langle \mathcal{A}, \mathcal{S}, \{\preccurlyeq_s : s \in \mathcal{S}\} \rangle$, as given in Definition 4.1, is a framework where $\mathcal{A}$ is a set of decisions, $\mathcal{S}$ is a set of possible scenarios, and for each $s \in \mathcal{S}$, relation $\preccurlyeq_s$ is a total preorder on $\mathcal{A}$. The work in Chapter 4 looks at the preference relations and notions of optimality that arise in this framework, and we will recall the relevant definitions

as we require in the next section.

6.3 Sorted-Pareto MODS

In this section, we look at the relationship betweeen Sorted-Pareto and Min-sum of weights as an instance of the MODS framework. Recall from Section 5.3, where for some ordinal scale T, we define F'_T as the set of all possible weight functions $f : T \to \mathbb{R}^+$ such that f is strictly monotonic with respect to T, i.e., for $u, v \in T$, $u \leq v \iff f(u) \leq f(v)$. The result in Theorem 5.2 gives us that for all $\alpha, \beta, \in \mathcal{A}$, $\alpha \preccurlyeq_{\text{SP}} \beta \iff \alpha \leq_f \beta$ for all $f \in F'_T$, i.e., we have that decision α Weak Sorted-Pareto dominates decision β if and only if α is min-sum-preferred to β for all $f \in F'_T$.

Given this result and the relationship between the Sorted-Pareto dominance relations and the Min-sum of weights relations we now define an instance of the MODS framework where we view a scenario $s \in \mathcal{S}$ as a choice of function $f \in F'_T$ from the min-sum case. i.e., we have a set of possible functions F'_T, with a corresponding set of orderings $\{\leq_f : f \in F'_T\}$. We call the *Sorted Pareto* MODS, and we define it as follows:

Definition 6.1 » Sorted-Pareto MODS $\mathcal{R}$
Let MODS $\mathcal{R} = \langle \mathcal{A}, F'_T, \{\leq_f : f \in F'_T\} \rangle$, where

- $\mathcal{A}$ is a non-empty, finite set of decisions,
- the set of scenarios is given by the set F'_T of strictly monotonic weight functions $f : T \to \mathbb{R}^+$,
- the set of orderings is given by the min-sum of weights orderings for all possible weight functions, i.e., the set $\{\leq_f : f \in F'_T\}$. «

We now look at the definitions for the MODS order relations used in the Sorted-Pareto MODS $\mathcal{R}$. From Section 4.2.1, we have the necessarily dominates relation $\preccurlyeq_{\text{N}}$, the strict part $\prec_{\text{N}}$ of the necessarily dominates relation, and the necessarily strictly dominates relation $\prec_{\text{NS}}$, which we now define in terms of the Sorted-Pareto MODS.

Definition 6.2 » Necessarily dominates for MODS $\mathcal{R}$
A decision α *necessarily dominates* β, written as $\alpha \preccurlyeq_{\text{N}} \beta$, if and only if

— $\alpha \leq_f \beta$ for all $f \in F'_T$ «

That is, for any choice of $f \in F'_T$, α is Min-sum-preferred to β.

Definition 6.3 » Strict-necessarily dominates for MODS $\mathcal{R}$
A decision α *strict necessarily dominates* β, written as $\alpha \prec_{\text{N}} \beta$, if and only if

— $\alpha \preccurlyeq_{\text{N}} \beta$ and $\beta \not\preccurlyeq_{\text{N}} \alpha$ «

Definition 6.4 » Necessarily strictly dominates for MODS $\mathcal{R}$
A decision α *necessarily strictly dominates* β, written as $\alpha \prec_{\text{NS}} \beta$, if and only if

— $\alpha <_f \beta$ for all $f \in F'_T$ «

That is, for any choice of f, α is strictly Min-sum-preferred to β.

We now give some results for the Sorted-Pareto MODS. Proposition 6.1 gives us that the necessarily dominates relation for MODS $\mathcal{R}$ corresponds to the Weak Sorted-Pareto dominance relation.

Proposition 6.1 » $\preccurlyeq_{\text{N}}$ for MODS $\mathcal{R}$ **result.**

For MODS $\mathcal{R}$, $\preccurlyeq_{\text{N}} = \preccurlyeq_{\text{SP}}$ ◇

Proof: This follows from the definitions and from Theorem 5.2. ∎

Our next result in Proposition 6.2 gives that the strict part of the necessarily dominates relation and the necessarily strictly dominates relation for MODS $\mathcal{R}$ are equal, and they correspond to the Sorted-Pareto dominance relation $\prec_{\text{SP}}$.

Proposition 6.2 » $\prec_{\text{NS}}$ and $\prec_{\text{N}}$ for MODS $\mathcal{R}$ **result**

For MODS $\mathcal{R}$, $\prec_{\text{NS}} = \prec_{\text{N}} = \prec_{\text{SP}}$ ◇

Proof: First we show that $\prec_{\text{N}} = \prec_{\text{SP}}$. From Proposition 6.1 we have that $\preccurlyeq_{\text{N}} = \preccurlyeq_{\text{SP}}$. Since by definition, we have that $\alpha \prec_{\text{SP}} \beta$ if and only if $\alpha \preccurlyeq_{\text{SP}} \beta$ and $\beta \not\preccurlyeq_{\text{SP}} \alpha$, and by definition of $\prec_{\text{N}}$ we have that $\alpha \prec_{\text{N}} \beta$ if and only if $\alpha \preccurlyeq_{\text{N}} \beta$ and $\beta \not\preccurlyeq_{\text{N}} \alpha$, then we have that $\prec_{\text{N}} = \prec_{\text{SP}}$.

Now we show that $\prec_{\text{SP}} = \prec_{\text{NS}}$. We have from Corollary 5.3 that $<_{\cap_{F'}} = \prec_{\text{SP}}$. By definition $<_{\cap_{F'}}$ is defined as the intersection of all $<_f$ such that $f \in F'_T$. So $\prec_{\text{NS}} = <_{\cap_{F'}}$. Therefore we have that $\prec_{\text{SP}} = \prec_{\text{NS}}$. ∎

Now we define an equivalence relation $\equiv_{F'}$, which is the intersection of $\equiv_f$ over all $f \in F'_T$, as follows:

Definition 6.5 » **F'-equivalent**
For all $\alpha, \beta \in \mathcal{A}$, $\alpha \equiv_{F'} \beta$ if and only if

— $\alpha \equiv_f \beta$, for all $f \in F'_T$ «

That is, α and β are F'-equivalent if they are equivalent over all possible choice of $f \in F'_T$. Given the definition of $\equiv_{F'}$, we define the *F'-equivalence* class of $\alpha \in \mathcal{A}$ as follows:

Definition 6.6 » **F'-equivalence class**
Let $[\alpha]_{F'} = \{\beta \in \mathcal{A} : \alpha \equiv_{F'} \beta\}$ «

The next result gives us that the necessarily equivalence relation for the Sorted-Pareto MODS $\mathcal{R}$ corresponds to Sorted-Pareto equivalence.

Proposition 6.3 » **$\equiv_{\text{N}}$ for** MODS $\mathcal{R}$ **result**

For MODS $\mathcal{R}$, $\equiv_{\text{N}} = \equiv_{\text{SP}}$ ◇

Proof: This follows from Proposition 6.1. ∎

6.3.1 Sorted-Pareto Optimality classes

We now look at the notions of optimality from the MODS framework that are applicable for the Sorted-Pareto MODS $\mathcal{R} = \langle \mathcal{A}, F'_T, \{\leq_f : f \in F'_T\} \rangle$. As before, α

and β are arbitrary elements of $\mathcal{A}$.

First we have the set of necessarily optimal elements for the Sorted-Pareto MODS $\mathcal{R}$, which are denoted by $\mathrm{NO}(\mathcal{R})$ and are defined as follows.

Definition 6.7 » Necessarily optimal for MODS $\mathcal{R}$
A decision α is in $\mathrm{NO}(\mathcal{R})$ if and only if

— for all $\beta \in \mathcal{A}$, for all $f \in F'_T$, $\alpha \leq_f \beta$. «

That is, we have that α is necessarily optimal if and only if for all $\beta \in \mathcal{A}$, for any choice of f, α is min-sum-preferred to β, which is if and only if $\alpha \preccurlyeq_{\mathrm{SP}} \beta$ for all $\beta \in \mathcal{A}$.

Next we have the set of necessarily strictly optimal elements for the Sorted-Pareto MODS, denoted by $\mathrm{NSO}(\mathcal{R})$, which are defined as follows.

Definition 6.8 » Necessarily strictly optimal for MODS $\mathcal{R}$
A decision α is in $\mathrm{NSO}(\mathcal{R})$ if and only if

— for all $\beta \in \mathcal{A} \setminus [\alpha]_{F'}$, for all $f \in F'_T$, $\alpha <_f \beta$. «

So we have that α is necessarily strictly optimal if and only if for any choice of f, α is strictly min-sum-preferred to every β not F'-equivalent to α, which is if and only if $\alpha \prec_{\mathrm{SP}} \beta$ for all $\beta \in \mathcal{A} \setminus [\alpha]_{\mathrm{SP}}$.

From these definitions, we have the following result.

Proposition 6.4 » $\mathrm{NSO}(\mathcal{R})$, $\mathrm{NOPSO}(\mathcal{R})$ and $\mathrm{NO}(\mathcal{R})$ result

For MODS $\mathcal{R}$, $\mathrm{NSO}(\mathcal{R}) = \mathrm{NOPSO}(\mathcal{R}) = \mathrm{NO}(\mathcal{R})$ ◇

Proof: This follows from the definitions of $\mathrm{NSO}(\mathcal{R})$ and $\mathrm{NO}(\mathcal{R})$, and from Proposition 6.2, where $\prec_{\mathrm{N}} = \prec_{\mathrm{NS}}$. ∎

We now look at definitions for the $\mathrm{CD}(\mathcal{R})$ and $\mathrm{CSD}(\mathcal{R})$ optimality classes for the Sorted-Pareto MODS.

Definition 6.9 » Can dominate for MODS $\mathcal{R}$
A decision α is in CD($\mathcal{R}$) if and only if

— for all $\beta \in \mathcal{A}$, there exists $f \in F'_T$ such that $\alpha \leq_f \beta$. «

That is, we have that decision α is in CD($\mathcal{R}$) if and only if it can be min-sum-preferred to any other decision.

Definition 6.10 » Can strictly dominate for MODS $\mathcal{R}$
A decision α is in CSD($\mathcal{R}$) if and only if

— for all $\beta \in \mathcal{A} \setminus [\alpha]_{F'}$, there exists $f \in F'_T$ such that $\alpha <_f \beta$. «

So we have that decision α is in CSD($\mathcal{R}$) if and only if it can be strictly min-sum-preferred to any non-equivalent decision. The decisions in the set CSD($\mathcal{R}$) are the decisions that are optimal or undominated with respect to the $\prec_{\mathrm{N}}$ relation, i.e., these are the Sorted-Pareto optimal decisions.

Proposition 6.5 » CSD($\mathcal{R}$) and CD($\mathcal{R}$) result

For MODS $\mathcal{R}$, CSD($\mathcal{R}$) = CD($\mathcal{R}$) ◇

Proof: This follows from Proposition 6.2, i.e., for MODS $\mathcal{R}$ we have that $\prec_{\mathrm{N}} = \prec_{\mathrm{NS}}$, and from the definitions of CD($\mathcal{G}$) and CSD($\mathcal{G}$) in the general case, where CD($\mathcal{G}$) are the decisions that are undominated with respect to $\prec_{\mathrm{NS}}$ and CSD($\mathcal{G}$) are the decisions that are undominated with respect to $\prec_{\mathrm{N}}$. ▪

Now we look at the possibly optimal and possibly strictly optimal decisions for the Sorted-Pareto MODS.

Definition 6.11 » Possibly optimal for MODS $\mathcal{R}$
A decision α is in PO($\mathcal{R}$) if and only if

— there exists $f \in F'_T$ such that for all $\beta \in \mathcal{A}$, $\alpha \leq_f \beta$. «

That is, α is in PO($\mathcal{R}$) if and only if there exists some choice of $f \in F'_T$ such that α is min-sum-optimal with respect to that f.

Definition 6.12 » Possibly strictly optimal for MODS $\mathcal{R}$
A decision α is in PSO($\mathcal{R}$) if and only if

— there exists $f \in F'_T$ such that for all $\beta \in \mathcal{A} \setminus [\alpha]_{F'}$, $\alpha \leq_f \beta$. «

That is, α is in PSO($\mathcal{R}$) if and only if there exists some choice of $f \in F'_T$ such that α is strictly min-sum-optimal with respect to that f.

Proposition 6.6 » PO($\mathcal{R}$) and CSD($\mathcal{R}$) result

For MODS $\mathcal{R}$, PO($\mathcal{R}$) $\subseteq$ CSD($\mathcal{R}$). ◇

Proof: This follows from Proposition 6.5, i,e., CSD($\mathcal{R}$) = CD($\mathcal{R}$), and from Proposition 4.5, where we have in the general case that PO($\mathcal{G}$) $\subseteq$ CD($\mathcal{G}$). ∎

6.3.2 Subclass relationships for Sorted-Pareto

Figure 6.1 shows precisely the subclass relationships between these optimality classes that always hold in the general case, which is given by Theorem 4.1; the theorem also gives an example of strict subclass relationships between each of the optimality classes.

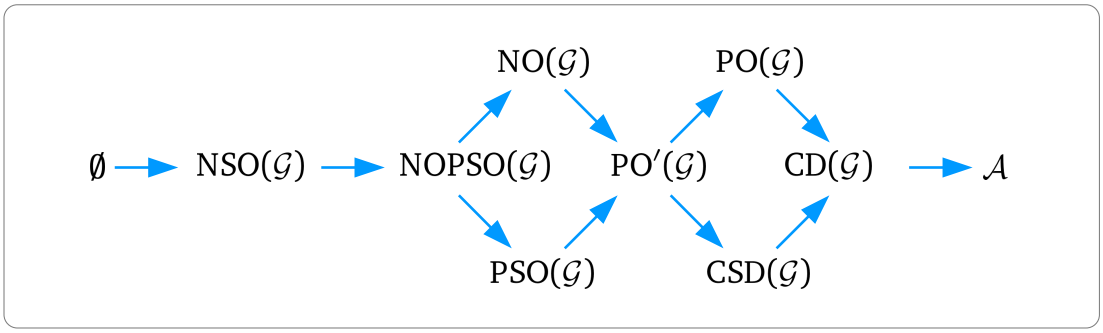


Figure 6.1: Subclass relationships ($\subseteq$) between optimality classes that always hold in general.

Given the results in the previous section, we now look at the subclass relationships between the optimality classes for the Sorted-Pareto and Min-sum instance of the MODS framework. The result below in Proposition 6.7 is also depicted in Figure 6.2.

Proposition 6.7 » Subclass relationships for Sorted-Pareto MODS

For MODS $\mathcal{R}$*, we have the following subclass relationships:*

$(\text{NSO}(\mathcal{R}) = \text{NOPSO}(\mathcal{R}) = \text{NO}(\mathcal{R})) \subseteq \text{PSO}(\mathcal{R}) \subseteq (\text{PO}(\mathcal{R}) = \text{PO}'(\mathcal{R})) \subseteq (\text{CSD}(\mathcal{R}) = \text{CD}(\mathcal{R}))$ ◇

Proof: This follows from Propositions 6.1 to 6.6 and Figure 6.1, which is adapted from Theorem 4.1 and shows the subclass relationships between the optimality classes that always hold in general for any MODS $\mathcal{G}$. ∎

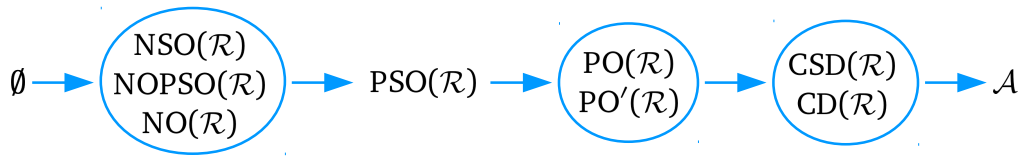


Figure 6.2: Subclass relationships ($\subseteq$) between the optimality classes for the Sorted-Pareto MODS $\mathcal{R}$.

Now we consider the case where there exists a decision that is necessarily optimal, i.e., when we have that $\text{NO}(\mathcal{R}) \neq \emptyset$.

Proposition 6.8 » Exists necessarily optimal element for Sorted-Pareto MODS **result**

For MODS $\mathcal{R}$*, if* $\text{NO}(\mathcal{R}) \neq \emptyset$*, then*

— $\text{NSO}(\mathcal{R}) = \text{NO}(\mathcal{R}) = \text{PSO}(\mathcal{R}) = \text{PO}(\mathcal{R}) = \text{CSD}(\mathcal{R}) = \text{CD}(\mathcal{R}) \subseteq \mathcal{A}$ ◇

Proof: This follows from: Proposition 4.8 (i), where we have that in the general case if $\text{NO}(\mathcal{G}) \neq \emptyset$ then $\text{NO}(\mathcal{G}) = \text{CSD}(\mathcal{G})$, and from Proposition 6.5 where we have for MODS $\mathcal{R}$ that $\text{CSD}(\mathcal{R}) = \text{CD}(\mathcal{R})$. ∎

6.4 Computing Optimality Classes for the Sorted-Pareto MODS

In this section, we look at methods for generating the optimality classes PO($\mathcal{R}$) and PSO($\mathcal{R}$) for the Sorted-Pareto MODS $\mathcal{R}$. Here we assume that there is some procedure to generate CSD($\mathcal{R}$), i.e., that calculates the preference vectors for each decision and compares them using Sorted-Pareto dominance to generate the set of decisions that are non-dominated. For example, the branch and bound search algorithms detailed in Section 5.4 do exactly this; however other search procedures can be used. From CSD($\mathcal{R}$), NO($\mathcal{R}$) can be calculated by comparing all the solutions in CSD($\mathcal{R}$) with one another to see if any Sorted-Pareto dominate all others.

First let us look at an example for PO($\mathcal{R}$) and PSO($\mathcal{R}$), where we have that PSO($\mathcal{R}$) $\subset$ PO($\mathcal{R}$).

Example 6.1 ► PSO($\mathcal{R}$) not equals PO($\mathcal{R}$) example**.**
Suppose for some problem, where the possible preference values are on an ordered scale of costs $T = \{A, B, C, D, E\}$*, and we have* $\mathcal{A} = \{\alpha, \beta, \gamma\}$*, with*

- $\rho(\alpha) = \{A, B, C, D, E\}$,
- $\rho(\beta) = \{A, C, C, D, D\}$,
- $\rho(\gamma) = \{A, B, B, E, E\}$.

We have that α*,* β *and* γ *are all undominated with respect to the Sorted-Pareto relation* $\prec_{\mathrm{N}}$*, so therefore we have that*

- CSD($\mathcal{R}$) $= \{\alpha, \beta, \gamma\}$.

Consider some possible weight functions, f_1 *and* f_2*, defined as follows, where the ordered pair* $\big(i, f(i)\big)$ *denotes the mapping from* $i \in T$ *to* $f(i) \in \mathbb{R}^+$*:*

- $f_1 = \{(A, 0), (B, 1), (C, 2), (D, 3), (E, 5)\}$
- $f_2 = \{(A, 0), (B, 1), (C, 3), (D, 4), (E, 5)\}$

For f_1*, we have that* β *is strictly min-sum optimal, i.e.,* $\beta \in$ PSO($\mathcal{R}$)*, since*

- $\sum_{i=i}^{m} f_1(\beta_i) = 10$,
- $\sum_{i=i}^{m} f_1(\alpha_i) = 11$,
- $\sum_{i=i}^{m} f_1(\gamma_i) = 12$,

and therefore $\beta <_{f_1} \alpha$ *and* $\beta <_{f_1} \gamma$.

For f_2*, we have that* γ *is strictly min-sum optimal, i.e.,* $\gamma \in \mathrm{PSO}(\mathcal{R})$*, since*

- $\sum_{i=i}^{m} f_2(\gamma_i) = 12$,
- $\sum_{i=i}^{m} f_2(\alpha_i) = 13$,
- $\sum_{i=i}^{m} f_2(\beta_i) = 14$,

and therefore $\gamma <_{f_2} \alpha$ *and* $\gamma <_{f_2} \beta$.

However there is no possible choice of weight function f *that would make* α *be strictly min-sum optimal for that* f.

We have that $f(\alpha) <_f f(\beta)$ *if and only if* $f(A) + f(B) + f(C) + f(D) + f(E) < f(A) + 2f(C) + 2f(D)$,

which is if and only if

- $f(B) + f(E) < f(C) + f(D)$

We also have that $f(\alpha) <_f f(\gamma)$ *if and only if* $f(A) + f(B) + f(C) + f(D) + f(E) < f(A) + 2f(B) + 2f(E)$,

which is if and only if

- $f(C) + f(D) < f(B) + f(E)$

Hence, we cannot have a function f *which has both* $f(\alpha) <_f f(\beta)$ *and* $f(\alpha) <_f f(\gamma)$*, so* $\alpha \notin \mathrm{PSO}(\mathcal{R})$.

We have however that α *is in* $\mathrm{PO}(\mathcal{R})$*, since, for example, for* f_3*, defined as*

- $f_3 = \{(A, 0), (B, 1), (C, 2), (D, 3), (E, 4)\}$

we have the following sum of weights for each decision

- $\sum_{i=i}^{m} f_2(\alpha_i) = 10$,
- $\sum_{i=i}^{m} f_2(\beta_i) = 10$,
- $\sum_{i=i}^{m} f_2(\gamma_i) = 10$

Therefore the resulting optimality classes are

- $\mathrm{PO}(\mathcal{R}) = \{\alpha, \beta, \gamma\}$
- $\mathrm{PSO}(\mathcal{R}) = \{\beta, \gamma\}$ ▲

6.4.1 Calculating Possibly Optimal

Suppose we want to determine if some decision α in CSD($\mathcal{R}$) is possibly optimal, i.e., we want to determine if there exists some weight function $f \in F$ such that $\alpha \leq_f \beta$ for all $\beta \in \text{CSD}(\mathcal{R})$. We can formulate this problem as a *linear program*. Firstly, we give a definition for a linear program as follows.

Definition 6.13 » Linear program
A *linear program* is a tuple $\langle \mathcal{X}, g, \mathcal{C} \rangle$, where,

- $\mathcal{X}$ is a set of problem or decision variables,
- g is an linear objective function (e.g., *maximise* or *minimise*), expressed in terms of the problem variables,
- $\mathcal{C}$ is a set of constraints on the variables, expressed in terms of linear equalities and linear inequalities using $\leq$ and $\geq$. «

A linear program solver can be used to solve the linear program and optimise the objective function.

To calculate if some decision α in CSD($\mathcal{R}$) is possibly optimal, then the problem can be formulated as a linear program P_α as follows. Only certain elements on the scale T appear in any of the preference vectors for the decisions in CSD($\mathcal{R}$); let T' denote this set, i.e., $T' = \{i \in v(\beta) : \beta \in \text{CSD}(\mathcal{R})\}$. For each of these elements $i \in T'$ we have a linear program variable w_i, representing an unknown weight. Since the scale T is totally ordered, then on these weights we have constraints of the form $w_i < w_j$, where $i < j$. For all $\beta \in \text{CSD}(\mathcal{R})$, we have a linear expression $\omega(\beta)$ as a sum in terms of the unknown weight variables, i.e., $\omega(\beta) = \sum_{i \in v(\beta)} w_i$. For α to be possibly optimal, we require, for each $\beta \in \text{CSD}(\mathcal{R})$, that $\omega(\alpha) \leq \omega(\beta)$. Therefore we have a set of linear inequalities, defined as follows:

Definition 6.14 » Set of linear inequalities P_α for PO($\mathcal{R}$)
Given some set CSD($\mathcal{R}$), to calculate if some $\alpha \in \text{CSD}(\mathcal{R})$ is possibly optimal, let P_α be a set of linear inequalities where we have:

(i) $w_i < w_j$, for all $i, j \in T'$, where $i < j$, and

(ii) $\omega(\alpha) \leq \omega(\beta)$, for all $\beta \in \text{CSD}(\mathcal{R})$. «

This gives us that if P_α has a feasible solution, then there exists some weights that make $\omega(\alpha) \leq \omega(\beta)$ for all $\beta \in \text{CSD}(\mathcal{R})$, i.e., we have that α is possibly optimal.

In order to check this using a standard linear program solver, we need to convert to an equivalent problem which only has non-strict inequalities, as we can see from Definition 6.13 that the inequality constraints need to be of the form $\leq$ or $\geq$. Therefore, we create a linear program as follows.

Definition 6.15 » Linear program P^c_α for $\text{PO}(\mathcal{R})$
To calculate if some $\alpha \in \text{CSD}(\mathcal{R})$ is possibly optimal, we create linear program P^c_α from the set of linear inequalities P_α, where $c > 0$ is some arbitrary strictly positive real number, e.g., we can set $c = 1$, and

(i) for any constraint in P_α in the form of Definition 6.14 (i), i.e., $w_i < w_j$, we have a constraint in P^c_α with the form $w_j - w_i \geq c$, and

(ii) for any constraint in P_α in the form of Definition 6.14 (ii), i.e., $\omega(\alpha) \leq \omega(\beta)$, we have a constraint in P^c_α of the form $\omega(\beta) - \omega(\alpha) \geq 0$. «

We then solve the linear program P^c_α, and this has a solution if and only if P_α has a solution, and therefore we have that α is possibly optimal. We have the following result.

Proposition 6.9 » Calculating $\text{PO}(\mathcal{R})$ result

For $\alpha \in \text{CSD}(\mathcal{R})$, with set of linear equalities P_α and linear program P^c_α with $c > 0$, we have that

— *P_α has a solution if and only if P^c_α has a solution.* ◇

Proof: First we show that for $\alpha \in \text{CSD}(\mathcal{R})$, if P_α has a solution, then P^c_α has a solution. Suppose that for $\alpha \in \mathcal{A}$, P_α has a solution s, where for all $w_i \in \{w_i : i \in T'\}$, we have $s(w_i) = w^s_i$, and for any $\alpha \in \text{CSD}(\mathcal{R})$, let $\omega^s(\alpha) = \sum_{i \in \upsilon(\alpha)} w^s_i$. Now let $\delta = \min_{i,j \in T', i<j} w^s_j - w^s_i$, and let $\lambda = \frac{c}{\delta}$. We define s' as a function such that for all $w_i \in \{w_i : i \in T'\}$, we have $s'(w_i) = \lambda w^s_i = w^{s'}_i$. Then we have for all $i, j \in T'$, where $i < j$, $w^{s'}_j - w^{s'}_i = \lambda(w^s_j - w^s_i) = \frac{c}{\delta}(w^s_j - w^s_i) \geq \frac{c}{\delta}\delta = c$. Thus we have for all $i, j \in T'$, where $i < j$, $w^{s'}_j - w^{s'}_i \geq c$. Since we have for solution s that for all $\beta \in \text{CSD}(\mathcal{R})$, $\sum_{i \in \upsilon(\alpha)} w^s_i \leq \sum_{i \in \upsilon(\beta)} w^s_i$, then we have for s' that for all $\beta \in \text{CSD}(\mathcal{R})$, $\sum_{i \in \upsilon(\beta)} w^{s'}_i - \sum_{i \in \upsilon(\alpha)} w^{s'}_i \geq 0$, i.e., $\omega^{s'}(\beta) - \omega^{s'}(\alpha) \geq 0$. Thus, s' is a solution for P^c_α.

Now we will show that, for $\alpha \in \text{CSD}(\mathcal{R})$, a solution of P^c_α is also a solution of P_α. For $\alpha \in \text{CSD}(\mathcal{R})$, suppose s is a solution for P^c_α. Then for all $\beta \in \text{CSD}(\mathcal{R})$, we have $\omega^s(\beta) - \omega^s(\alpha) \geq 0$, so we have $\omega^s(\alpha) \leq \omega^s(\beta)$. We also have for all $i, j \in T'$, where $i < j$, $w^s_j - w^s_i \geq c$. Since $c > 0$, then for all $i, j \in T'$, where $i < j$, we have $w^s_i < w^s_j$. Therefore s is a solution for P_α. ∎

6.4.2 Calculating Possibly Strictly Optimal

We can also determine if some decision in $\text{CSD}(\mathcal{R})$ is possibly strictly optimal, i.e., there exists f such that for all $\beta \in \text{CSD}(\mathcal{R}) \setminus [\alpha]$, $\alpha <_f \beta$. For some $\alpha \in \text{CSD}(\mathcal{R})$, we have a set Q_α of linear inequalities which is defined as follows.

Definition 6.16 » Set of linear inequalities Q_α for PSO($\mathcal{R}$)
To calculate if some $\alpha \in \text{CSD}(\mathcal{R})$ is possibly optimal, let Q_α be a set of linear inequalities where we have:

(i) $w_i < w_j$, for all $i, j \in T'$, where $i < j$ and,

(ii) $\omega(\alpha) < \omega(\beta)$, for all $\beta \in \text{CSD}(\mathcal{R})$. «

Then if Q_α has a feasible solution, then there exists some weights that make $\alpha < \beta$ for all $\beta \in \text{CSD}(\mathcal{R})$, i.e., α is possibly strictly optimal.

To solve this, we again create a linear program Q^c_α as follows:

Definition 6.17 » Linear program Q^c_α for PSO($\mathcal{R}$)
To calculate if some $\alpha \in \text{CSD}(\mathcal{R})$ is possibly strictly optimal, we create linear program Q^c_α from set of linear inequalities Q_α, where $c > 0$ is some arbitrary strictly positive real number, e.g., we can set $c = 1$, and

(i) for any constraint in Q_α in the form of Definition 6.16 (i), i.e., $w_i < w_j$, we have a constraint in Q^c_α with the form $w_j - w_i \geq c$, and

(ii) for any constraint in Q_α in the form of Definition 6.16 (ii), i.e., $\omega(\alpha) < \omega(\beta)$, we have a constraint in Q^c_α of the form $\omega(\beta) - \omega(\alpha) \geq c$. «

We then solve the linear program Q^c_α, and this has a solution if and only if Q_α has a solution, which gives us that α is possibly strictly optimal. We have the following result.

Proposition 6.10 » Calculating PSO($\mathcal{R}$) result

For $\alpha \in \text{CSD}(\mathcal{R})$, with set of linear equalities Q_α and linear program Q^c_α with $c > 0$, we have that

— *Q_α has a solution if and only if Q^c_α has a solution.* ◇

Proof: This is proved in a similar way to Proposition 6.9. First we show that for $\alpha \in \text{CSD}(\mathcal{R})$, if Q_α has a solution, then Q^c_α has a solution. Suppose that for $\alpha \in \text{CSD}(\mathcal{R})$, Q_α has a solution s. Now let $\delta_1 = \min_{i,j \in T', i<j} w^s_j - w^s_i$, and let $\delta_2 = \min_{\beta \in \text{CSD}(\mathcal{R})} \omega^s(\beta) - \omega^s(\alpha)$. Now we set $\delta = \min(\delta_1, \delta_2)$, and as before, let $\lambda = \frac{c}{\delta}$. We define s' as a function such that for all $w_i \in \{w_i : i \in T'\}$, we have $s'(w_i) = \lambda w^s_i = w^{s'}_i$. Then we have for all $i, j \in T'$, where $i < j$, $w^{s'}_j - w^{s'}_i = \lambda(w^s_j - w^s_i) = \frac{c}{\delta}(w^s_j - w^s_i) \geq \frac{c}{\delta}\delta = c$. Thus we have for all $i, j \in T'$, where $i < j$, $w^{s'}_j - w^{s'}_i \geq c$. Since we have for solution s that for all $\beta \in \text{CSD}(\mathcal{R})$, $\sum_{i \in \upsilon(\alpha)} w^s_i < \sum_{i \in \upsilon(\beta)} w^s_i$, and since for all $i, j \in T'$, where $i < j$, we have $w^{s'}_j - w^{s'}_i \geq c$ then we have for s' that for all $\beta \in \text{CSD}(\mathcal{R})$, $\sum_{i \in \upsilon(\beta)} w^{s'}_i - \sum_{i \in \upsilon(\alpha)} w^{s'}_i \geq c$, i.e., $\omega^{s'}(\beta) - \omega^{s'}(\alpha) \geq 0$. Thus, s' is a solution for P^c_α.

Now we will show that, for $\alpha \in \text{CSD}(\mathcal{R})$, a solution of Q^c_α is also a solution of Q_α. For $\alpha \in \text{CSD}(\mathcal{R})$, suppose s is a solution for Q^c_α. Then for all $\beta \in \text{CSD}(\mathcal{R})$, we have $\omega^s(\beta) - \omega^s(\alpha) \geq c$, so we have $\omega^s(\alpha) \leq \omega^s(\beta)$. We also have for all $i, j \in T'$, where $i < j$, $w^s_j - w^s_i \geq c$. Since $c > 0$, then for all $i, j \in T'$, where $i < j$, we have $w^s_i < w^s_j$. Therefore s is a solution for Q_α. ∎

6.5 Experimental Results

In this section, we calculate the optimality classes CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$), and NO($\mathcal{R}$) for some randomly generated and benchmark constraint problem instances. To solve these problems, we use the soft constraints solver implementation as detailed in Section 5.6, which was extended to calculate the MODS optimality classes for the Sorted-Pareto MODS $\mathcal{R}$. The branch and bound algorithms from Section 5.4 are used to generate CSD($\mathcal{R}$) and NO($\mathcal{R}$), and to generate PO($\mathcal{R}$) and PSO($\mathcal{R}$) the solver creates and solves linear programs, as detailed in Section 6.4.

In the tables of results, we use the following notation for each optimality class. We

let CSD denote the set CSD($\mathcal{R}$), and we let $\text{CSD}_{[]}$ denote the set of equivalence classes of CSD($\mathcal{R}$). To relate this notation to that used in the results in Section 5.7, CSD is the same as O_{SP}, and $\text{CSD}_{[]}$ is the same as $\text{O}_{[\text{SP}]}$, since we have that the Sorted-Pareto optimal decisions O_{SP} are those that are undominated with respect to $\prec_{\text{N}}$, as given in Proposition 4.2. Similarly we let PO denote the set PO($\mathcal{R}$) and $\text{PO}_{[]}$ the set of equivalences classes of PO($\mathcal{R}$), and so on.

6.5.1 Random Problems

In this section, we look at the results for some of the sets of random problem instances as used in the experimental result section in Chapter 5. Table 6.1 shows the average size of the optimality classes and the average number of equivalence classes for each of CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$) and NO($\mathcal{R}$), for the instances in Set B (see Table 5.1), where there are 50 random instances for each of problem size $n = 20, 24, \ldots, 40$. For NO($\mathcal{R}$), in the table we show the number of instances where NO($\mathcal{R}$) is not empty (NO $\neq \emptyset$), and the average number of solutions given for NO($\mathcal{R}$) is over the number of non-empty instances, denoted by $\langle\text{NO}\rangle^*$.

In these problems, it can be observed that PO($\mathcal{R}$) is usually smaller than CSD($\mathcal{R}$), with PSO($\mathcal{R}$) usually smaller again. In nearly all cases NO($\mathcal{R}$) is empty, so in these cases there does not exists a necessarily optimal solution.

Table 6.1: The average size of the optimality classes and the average number of equivalence classes for each of CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$) and the number of instances where NO($\mathcal{R}$) is non-empty, for the randomly generated problems in Set B, where there are 50 random instances for each of $n = 20, 24, \ldots, 40$.

Set B	$n = 20$	$n = 24$	$n = 28$	$n = 32$	$n = 36$	$n = 40$
$\langle\text{Sol}\rangle$	44260.48	125977.84	356328.00	764423.20	1218371.52	1462344.32
$\langle\text{CSD}\rangle$	50.90	89.12	141.18	154.26	316.20	335.18
$\langle\text{CSD}_{[]}\rangle$	18.66	21.32	16.62	13.90	11.22	9.58
$\langle\text{PO}\rangle$	44.74	76.48	120.30	142.62	304.80	309.22
$\langle\text{PO}_{[]}\rangle$	16.84	18.34	14.64	12.66	10.48	9.02
$\langle\text{PSO}\rangle$	43.90	73.68	120.26	142.62	304.80	309.22
$\langle\text{PSO}_{[]}\rangle$	16.68	18.22	14.62	12.66	10.48	9.02
NO $\neq \emptyset$	1/50	0/50	0/50	1/50	0/50	1/50
$\langle\text{NO}\rangle^*$	4	-	-	3	-	64

* average over instances where NO is non-empty

Table 6.2 shows the average size of the optimality classes and the average number of equivalence classes for each of CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$), for the instances in Set C (see Table 5.1), where there are 50 random instances for each of $sc = 5, 10, \ldots, 30$.

Again for the NO($\mathcal{R}$) case, we show the number of instances where NO($\mathcal{R}$) is not empty (NO $\neq \emptyset$), and the average number of solutions over the non-empty instances, denoted by $\langle \text{NO} \rangle^*$.

Often NO($\mathcal{R}$) is non-empty, indicating a single equivalence class of necessarily optimal solutions, and in these cases we have CSD($\mathcal{R}$) = PO($\mathcal{R}$) = PSO($\mathcal{R}$) = NO($\mathcal{R}$), by Proposition 6.8.

Table 6.2: The average size of the optimality classes and the average number of equivalence classes for each of CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$), and the number of instances where NO($\mathcal{R}$) is non-empty, for the randomly generated problems in Set C, where there are 50 random instances for each of $sc = 5, 10, \ldots, 30$.

Set C	$sc = 5$	$sc = 10$	$sc = 15$	$sc = 20$	$sc = 25$	$sc = 30$
$\langle$Sol$\rangle$	188787.52	188787.52	188787.52	188787.52	188787.52	188787.52
$\langle$CSD$\rangle$	8470.96	1084.80	214.08	84.24	53.90	24.10
$\langle \text{CSD}_{[]} \rangle$	1.10	1.32	1.56	1.84	2.24	2.30
$\langle$PO$\rangle$	8470.96	1084.80	211.68	81.08	53.90	21.70
$\langle \text{PO}_{[]} \rangle$	1.10	1.32	1.54	1.82	2.24	2.18
$\langle$PSO$\rangle$	8470.96	1084.80	204.96	72.80	40.08	20.48
$\langle \text{PSO}_{[]} \rangle$	1.10	1.32	1.48	1.66	2.00	2.02
NO $\neq \emptyset$	45/50	34/50	27/50	21/50	8/50	13/50
$\langle \text{NO} \rangle^*$	8170.40	870.71	178.89	41.10	10.50	5.77

* average over instances where NO is non-empty

6.5.2 Non-random Problems

Table 6.3 shows the optimality classes CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$) and NO($\mathcal{R}$) for the modified WCSP instances from the CELAR Radio-Link Frequency Assignment problem (RLFAP problem) [Schb] used in Section 5.7.4.

In these benchmark instances, PO($\mathcal{R}$) is usually smaller than CSD($\mathcal{R}$), but there are no instances in which PSO($\mathcal{R}$) is smaller than PO($\mathcal{R}$). In all of these instances, NO($\mathcal{R}$) is empty.

Table 6.3: The size of the optimality classes and the number of equivalence classes for each of CSD($\mathcal{R}$), PO($\mathcal{R}$), PSO($\mathcal{R}$), and NO($\mathcal{R}$) for the modified instances from the CELAR Radio-Link Frequency Assignment problem benchmark, where hard constraints have been added to limit the number of solutions to around 100,000.

Name	|Sol|	|CSD|	$\lvert CSD_{[]}\rvert$	|PO|	$\lvert PO_{[]}\rvert$	|PSO|	$\lvert PSO_{[]}\rvert$	|NO|	$\lvert NO_{[]}\rvert$
CELAR6-SUB0*	101660	17	9	16	8	16	8	0	0
CELAR6-SUB1*	91562	35	32	13	13	13	13	0	0
CELAR6-SUB2*	100783	20	20	11	11	11	11	0	0
CELAR6-SUB3*	96611	23	19	21	17	21	17	0	0
CELAR6-SUB4*	91994	36	27	15	13	15	13	0	0
CELAR7-SUB0*	91010	11	9	8	6	8	6	0	0
CELAR7-SUB1*	97437	19	17	12	10	12	10	0	0
CELAR7-SUB2*	93569	31	29	16	16	16	16	0	0
CELAR7-SUB3*	101851	42	34	22	17	22	17	0	0
CELAR7-SUB4*	97185	26	23	15	12	15	12	0	0

6.6 Discussion

One possible approach to choosing which decisions to present to a decision maker is to calculate CSD($\mathcal{R}$) first, and from this set, NO($\mathcal{R}$) can be easily derived. If NO($\mathcal{R}$) is not empty, then there are one or more equivalent decisions which are preferred to all other decisions for any choice of function $f : T \to \mathbb{R}^+$, and these are prime candidates for presenting to a decisions maker.

However, if NO($\mathcal{R}$) is empty, then PO($\mathcal{R}$) or PSO($\mathcal{R}$) can be computed and presented, these sets are often much smaller than CSD($\mathcal{R}$). PO($\mathcal{R}$) is the set of decisions that are min-sum optimal for some possible choice of function $f : T \to \mathbb{R}^+$, and thus are good candidates to present to a decision maker. If the PO($\mathcal{R}$) set is large, and there is a small number of equivalence classes, then a representative solution for each equivalence class could be chosen to present to a decision maker, since this would give a decision maker a choice between non-equivalent solutions that are possibly min-sum-optimal.

6.7 Chapter Conclusion

In this chapter, we looked at an instance of the multiple-ordering decision structure framework from Chapter 4 which captures the relationship between the Sorted-

Pareto dominance relation and Min-sum of weights relation as given in Section 5.3, and we examined the notions of optimality from the framework that are applicable to the Sorted-Pareto case. Specifically, we look at decisions that are undominated, i.e, CSD($\mathcal{R}$), the solutions that are optimal and strictly optimal in one (or more) scenarios, i.e., PO($\mathcal{R}$) and PSO($\mathcal{R}$) respectively, and the solutions that are optimal in all scenarios, i.e., NO($\mathcal{R}$). We explore the relations between these notions of optimality and show how to compute them for the Sorted-Pareto ordering and the min-sum of weights case. The experimental results show, that in some cases, NO($\mathcal{R}$) is non-empty, and these are the decisions that would be of most interest to a decision maker. However, in other cases, no such decisions exist, and then PO($\mathcal{R}$) and PSO($\mathcal{R}$) are of interest to a decision maker since these are the decisions that are optimal or strictly optimal in some scenario.

As discussed in the introduction to Sorted-Pareto dominance in Section 2.6.5, and in the given semantics for the relation in Section 5.3, Sorted-Pareto dominance connects with weighted constraint satisfaction problems (WCSP), or similarly, with generalised additive independence (GAI) decompositions, where a problem has only weights on an ordinal scale T; each such problem has a set of compatible proper weighted constraints problems, based on mapping the ordinal scale $T \rightarrow \mathbb{R}^+$. Sorted-Pareto is also connected to Bayesian Networks, where in a given network we only have ordinal probabilistic information and therefore we have an associated set of compatible Bayesian Networks. In a Weighted CSP with ordinal weights, the decisions that are possibly optimal, PO($\mathcal{R}$), are those that are min-sum optimal in some compatible weighted constraints problem, and in a Bayesian Network with ordinal probabilities, the possibly optimal decisions are those assignments that are most probable in some compatible Bayesian Network.

Chapter 7

Conclusion

7.1 Introduction

In this chapter, we conclude the thesis. The outline of the chapter is as follows. In Section 7.2 we look at a couple of possible future directions for some of the material in the thesis. The first is an extension of Sorted-Pareto dominance that also considers the relative importance of the decision aspects, and the second is a framework for uncertain softness soft constraints. In Section 7.3 we give a brief summary of the work, and some final remarks.

7.2 Possible Future Work

In this section, we look at some possible future directions for the work in this thesis. The first possible future work is an extension to Sorted-Pareto dominance relation which takes into account the relative importance of the decision aspects. The second is a framework for uncertain soft constraints incorporating Sorted-Pareto constraints problems and interval-valued constraints problems (see Section 4.6.2).

7.2.1 Lex-Sorted-Pareto Dominance

First we consider another extension to the Sorted-Pareto dominance relation. Consider for some multi-aspect decision problem $\langle \mathcal{A}, \mathcal{S}, T, \leq \rangle$, there is further ordinal information available on the importance of the aspects, for example, representing a situation where the aspects might be states that are much more likely to occur, or criteria that are much more important. This is similar to the approach taken by [Jun04] for handling preferences between criteria in multi-criteria problems, and to Lexicographic Constraint Satisifaction Problems in [FHWW10].

We give a definition of Lex-Sorted-Pareto dominance as follows. First, let $\mathcal{L} = \{L_1, \ldots, L_k\}$ be an ordered partition on the set of aspects $\mathcal{S}$, where each $i \in \mathcal{S}$ appears in only one $L \in \mathcal{L}$. The Lex-Sorted preference vector $v^{\uparrow}_{\mathcal{L}}(\alpha)$ of a decision $\alpha \in \mathcal{A}$ is given by,

$$v^{\uparrow}_{\mathcal{L}}(\alpha) = (v^{\uparrow}_{L_1}(\alpha), v^{\uparrow}_{L_2}(\alpha), \ldots, v^{\uparrow}_{L_k}(\alpha))$$

where each $v^{\uparrow}_{L_i}(\alpha)$ is the sub-vector of $v(\alpha)$ consisting of the preference values of the aspects in L_i. Then a decision Lex-Sorted-Pareto dominates another if and only if there exists $j \in \{1, \ldots, k\}$ such that for all $i < j$, $v^{\uparrow}_{L_i}(\alpha) = v^{\uparrow}_{L_i}(\beta)$, and

$v^{\uparrow}_{L_j}(\alpha) < v^{\uparrow}_{L_j}(\beta)$. This means that for the two decisions being compared, the sub-vector of the most important preferences are compared first, and if these sub-vectors are equal, then the preferences of the next most important aspects are considered, and so on.

In the general constraints problems considered in this thesis, there is no ordering or importance given to the soft constraints, i.e., we have no partition $\mathcal{L}$ of the decision aspects $\mathcal{S}$, so a possible future direction of the work in this thesis would be the extension of the constraints framework to include this sort of importance information, for which a preference relation like Lex-Sorted-Pareto dominance could then be implemented.

7.2.2 Uncertain Soft Constraints

As we saw in Chapters 5 and 6, the Sorted-Pareto preference relation can be used in situations of decision making under uncertainty, where we are uncertain about the actual numerical preference values but where instead we have a totally ordered set of qualitative preference degrees, for example, for a qualitative scale of costs we have a *low* cost is preferred to a *high* cost. Interval valued soft constraints [GPR$^+$10b], as seen in Section 4.6.2, associate a *preference interval* rather than a single preference level to an assignment, which caters for decision making where there may be some imprecision or uncertainty around the actual numerical value and instead allows a range of preference values to be specified.

We propose in brief, as a possible area for future work, the development of a soft constraints framework which captures constraints that associate either a qualitative preference level (such as in the Sorted-Pareto constraints problem) or an interval valued preference level (such as in Interval valued soft constraints) to variable assignments or tuples in a given problem. Then the different notions of optimality discussed in Chapters 4 and 6, and those in [GPR$^+$10b], would then apply. This approach described is also related to Stochastic Constraint Programming [Wal02] and Mixed Constraint Satisfaction [FLS96]. The future work in this area could involve the development of such a framework, and the implementation of different algorithms, such as in the spirit of those that are described in Section 5.4, to solve such Uncertain Soft Constraints problems.

7.3 Final Remarks

In a decision making problem, a preference relation is an ordering over the set of decisions, and when this ordering is not complete, we cannot compare each decision with one another. Such situations occur where there are multiple criteria in which to evaluate the decisions, or different possible states of the world to consider. In other situations we may only have qualitative (rather than quantitative) information, and sometimes this is because qualitative preferential information can be more reliable or easier to elicit than quantitative information. In any such case, we may end up with an ordering on the set of decisions which is only a preorder or a partial order, and we could have no single best decision. In Section 1.1, we introduced the thesis by asking what we should do in these decision making situations where there is not a single *best* decision. When we are supporting a decision maker, how do we choose what decisions to present to the decision maker? In the remainder of this section we look back over the work in the thesis to try and answer this question.

In Chapter 2, we gave definitions for some of the different types of preference or preorder relations, and described some of the properties of these relations. We described the notions of strict preference, where a decision is strictly preferred to another, and equivalence, where two decisions are equally preferred. The *best* decisions are the ones that are preferred to (or dominate) all other decisions, and in the case where a preference relation is not complete, the *optimal* (or undominated) decisions are the ones such that there are no other decisions that are strictly preferred to them. We looked at an extension to a preference relation, and we saw that extending the strict part of a preference relation resulted in a smaller optimal set for that extension.

We considered decision making problems where we have multiple decision aspects resulting in multiple preference values for each decision, i.e., where we have a vector of preference values for each decision. In such situations, the Pareto dominance relation is often used; it prefers decisions that are at least as good in every aspect, and strictly better in at least one aspect, and therefore, decisions that are Pareto optimal are desirable. The Sorted-Pareto dominance relation, which extends Pareto dominance, can be used in qualitative decision making situation where we can make the preference scales in each aspect commensurate, and where the ordering of the decision aspects (and resulting preference vectors) does not matter. Since Sorted-Pareto dominance extends Pareto dominance, then the resulting Sorted-Pareto optimal set for any problem is a subset of the Pareto optimal set.

We explored the connection between Sorted-Pareto dominance and other preference relations, such as Minimax, Lexicographic-Max ordering, Maximin, and Leximin, which are qualitative preference relations that also use a reordering of the preference vectors. We saw that these orderings can ignore preference values, known as "the drowning effect", and that Sorted-Pareto dominance considers all the values in the preference vectors when comparing decisions, which could be important in the context of supporting a decision maker and presenting decisions such that any preference values are not ignored. We also compared Sorted-Pareto to other preference relations which rely on quantitative information, such as relations that rely on a sum of weights, ordered weighted averages, and generalised Lorenz dominance. If the information available is of a quantitative nature then we could use such preference relations, however Sorted-Pareto only requires a qualitative scale and often this information is easier to elicit or more reliable.

In Chapter 3, we looked at hard and soft constraints, which are a natural way to model a decision making problem. A constraints problem specifies a set of variables, and the relationships between the domain values of the variables are modelled with constraints. The hard constraints specify the combination of domain values (or tuples) that are allowed, and the soft constraints associate preference values to tuples. Therefore we have a collection of preference values associated with each decision or *solution* to the problem, and as such we can use a soft constraints framework to model our decision making problem.

We looked at different constraints formalisms for hard and soft constraints, such as general frameworks like semiring constraints framework, and more specific frameworks such as weighted constraints satisfaction problems, and we looked at the preference relations associated with each formalism and the notion of optimal solutions for constraints networks. We also described some methods on how to solve constraints problem, focusing mainly on depth first search and depth first branch and bound search, algorithms which generate a set of best or optimal solutions to the problem. We looked at some consistency techniques which remove domain values that are inconsistent with the hard constraints of the problem, and we looked at how to generate a lower bound preference level for the soft constraints to help eliminate unnecessary dominance checks.

In Chapter 4, we looked further at the notion of *optimal* in decision making problems. We considered a decision making situation where we only assume qualitative information, i.e., we do not assume quantitative values or that we have a preference vector for each decision. In this setting, we have a set of decisions, and multiple

scenarios which each give a total preorder on the set of decisions, i.e., a decision is preferred, strictly preferred, or equivalent to every other decision in a scenario, and we called this a multiple-ordering decision structure (MODS). In this structure, we defined preference relations where a decision dominates (or strictly dominates) another decision in all scenarios, and decisions that are equivalent in all scenarios.

In the MODS framework, we investigated the different natural optimality classes that occur in this setup:

- decisions that dominate (or strictly dominate) all others in *all* scenarios, i.e., necessarily optimal (or necessarily strictly optimal),
- decisions that dominate (or strictly dominate) all others in *some* scenario, i.e., possibly optimal (or possibly strictly optimal),
- decisions that for every other decision there exists a scenario in which it can dominate (or can strictly dominate) that decision,
- decisions that are optimal in a maximal set of scenarios,
- decisions that are extreme decisions.

We provided some discussion on each of these optimality class, for example, the necessarily optimal decisions are the ones that are the optimal in every scenario, but often this class will be empty. In Theorem 4.1 we gave the precise subclass relationships between these classes. We also looked at how the subclass relationships simplified under additional conditions. When there exists a necessarily optimal decision, then the subclass relationships collapse to a chain. Given the relationships between these optimality classes, the set of decisions can be organised in such a way that they are categorised according to the minimal class in which they belong, which could be a useful information to present to a decision maker.

In Chapter 5, we looked further at the Sorted-Pareto dominance preference relation, giving some further properties. As well as considering Sorted-Pareto as a preorder on a set of decisions in a decision problem, we gave a characterisation of Sorted-Pareto as a partial order on multisets of preference values. The Sorted-Pareto dominance relation is unaffected by permutations, i.e., the orderings of the preference vector is irrelevant, and this is important in the context of soft constraints, where we can view the input as being a multiset of soft constraints. We also gave an important semantics for Sorted-Pareto dominance, as a relation that is consistent with any choice of weights function that maps an ordinal scale to a numerical one. We defined the Sorted-Pareto dominance constraints problem, based on the soft constraints

framework given in Chapter 3.3, and we looked at some different algorithms for generating a set of Sorted-Pareto optimal solutions. We looked at a naive generate and test algorithm, a depth first branch and bound algorithm with lower bound and a depth first branch and bound algorithm with an upper bound. We also looked at how equivalent solutions could be handled within the algorithm, instead of checking if a solution is dominated by each solution in the optimal set, the algorithm can maintain a map of preference levels to undominated solutions, where we have a single preference level mapping to a set of equivalence solutions.

The experimental results in Chapter 5 show that for a given set of problems the Sorted-Pareto optimal set of solutions is much smaller than the Pareto optimal set. The equivalence handing algorithms perform much better than the set-based algorithms, and the performance is related to the average sizes of the equivalence classes. We also saw that the size of the preference scale and the number of soft constraints in the problem affected the resulting number of Sorted-Pareto optimal solutions. We performed a comparison between the sizes of the Sorted-Pareto optimal solution sets and the Lexicographic-Max optimal solution sets; as a total preorder the Lexicographic-Max optimal sets were much smaller, but as seen in the comparision between Sorted-Pareto and Lexicographic-Max, the Lexicographic-Max preference relation can ignore preference values, so we may not want to present these to a decision maker if our decision maker is concerned about considering all preference values. For a smaller set of optimal solutions we saw that the Minmax-Sorted-Pareto optimal set is smaller than the Sorted-Pareto optimal set, and these solutions do not ignore any preference values.

In Chapter 6 we returned to the MODS framework from Chapter 4. Given the semantics for Sorted-Pareto in terms of a relation that is consistent with any choice of weights function that maps an ordinal scale to a numerical one as seen in Theorem 5.2, we looked at an instance of the MODS framework which captures this semantics, where a scenario corresponds to a choice of weights function. We looked at the optimality classes that apply to the Sorted-Pareto MODS:

- decisions that are min-sum optimal for any choice of f (NO($\mathcal{R}$))
- decisions that are strictly min-sum optimal for some choice of f (PSO($\mathcal{R}$)),
- decisions that are min-sum optimal for some choice of f (PO($\mathcal{R}$)),
- decisions that are min-sum preferred to every other decision in some choice of f (CSD($\mathcal{R}$)).

Given that we can calculate CSD($\mathcal{R}$) using our branch and bound algorithm for generating the Sorted-Pareto optimal set, we looked at how to calculate PSO($\mathcal{R}$) and PO($\mathcal{R}$) for the Sorted-Pareto MODS, using a linear program solver. The experimental results showed that NO($\mathcal{R}$) is often empty, but for some instances a necessarily optimal solution did exist. The possibly optimal solution set PO($\mathcal{R}$) was often much smaller than CSD($\mathcal{R}$), and these solutions would be good candidates to present to a decision maker, since they are possibly min-sum optimal for some choice of f.

In summary, in this thesis, we have looked at some different notions of optimality in qualitative and partially ordered decision making settings. We have shown that the Sorted-Pareto dominance relation is a natural preference ordering and we have provided some different characterisations of it. We have shown how it can be applied and computed in a soft constraints setting, and we have analysed theoretically and experimentally different notions of optimal solution in this context. We would argue that these notions of optimal solution for Sorted-Pareto would be of interest when choosing decisions to present to a decision maker.

The work in the thesis forms a framework for qualitative decision making, and situations and applications where we have only qualitative information can often naturally occur in decision making and decision support, for example, recommender systems. We argue that in these situations, our framework provides a useful presentation of the different notions of qualitative optimality for a decision maker, and by using the Sorted-Pareto dominance relation and computing associated optimality classes, this gives a decision maker some appropriate information in order to aid their choice.

The End.

References

[AH72] Kenneth J. Arrow and Leonid Hurwicz. *An Optimality Criterion for Decision Making under Ignorance*. Basil Blackwell, Oxford, 1972.

[Atk70] Anthony B. Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244–263, September 1970.

[BEN04] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve, Open source (Mixed-Integer) Linear Programming system. Software, May 1 2004. Available at http://lpsolve.sourceforge.net/5.5/.

[BG95] Fahiem Bacchus and Adam J. Grove. Graphical models for preference and utility. In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, (UAI 1995)*, 1995.

[BJ88] Salvador Barbarà and Matthew Jackson. Maximin, leximin, and the protective criterion: Characterizations and comparisons. *Journal of Economic Theory*, 46(1):34–44, October 1988.

[BL09] Sylvain Bouveret and Michel Lemaître. Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364, 2009.

[BMR97] Stefano Bistarelli, Ugo Montanari, and Francesca Rossi. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44:201–236, 1997.

[BMR+99] Stefano Bistarelli, Ugo Montanari, Francesca Rossi, Thomas Schiex, Gérard Verfaillie, and Hélène Fargier. Semiring-based CSPs and valued CSPs: Frameworks, properties, and comparison. *Constraints*, 4:199–240, 1999.

[BPPS06] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8-9):686 – 713, 2006.

[BR96] Christian Bessière and Jean-Charles Régin. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems. In *Proceedings of the 2nd International Conference on the Principles and Practices of Constraint Programming, (CP 1996)*, 1996.

[BS06] Ulrike Bossong and Dietmar Schweigert. Minimal paths on ordered graphs. *Mathematica Slovaca*, 56:23–31, 2006.

[BT97] Ronen I. Brafman and Moshe Tennenholtz. On the axiomatization of qualitative decision criteria. In *Journal of the ACM*, pages 76–81, 1997.

[CdGL+99] Bertrand Cabon, Simon de Givry, Lionel Lobjois, Thomas Schiex, and Joost P. Warners. Radio link frequency assignment. *Constraints*, 4:79–89, 1999.

[Cen13] Cork Constraint Computation Centre. 4C Cluster, Sep 2013. http://4c132.ucc.ie/index.php.

[CM10] Ronan Congar and François Maniquet. A trichotomy of attitudes for decision-making under complete ignorance. *Mathematical Social Sciences*, 59(1):15–25, 2010.

[CS04] Martin Cooper and Thomas Schiex. Arc consistency for soft constraints. *Artificial Intelligence*, 154:199 – 227, 2004.

[Dec99] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[Dec03] Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[DF98] Rina Dechter and Daniel Frost. Backtracking algorithms for constraint satisfaction problems - a survey. Technical report, Department of Information and Computer Science, University of California, Irvine, Irvine, California, USA, 1998.

[DF05] Didier Dubois and Philippe Fortemps. Selecting preferred solutions in the minimax approach to dynamic programming problems under flexible constraints. *European Journal of Operational Research*, 160(3):582–598, 2005.

[DFP96a] Didier Dubois, Hélène Fargier, and Henri Prade. Possibility Theory in Constraint Satisfaction Problems: Handling Priority, Preference and Uncertainty. *Applied Intelligence*, 6(4):287–309, 1996.

[DFP96b] Didier Dubois, Hélène Fargier, and Henri Prade. Refinements of the maximin approach to decision-making in a fuzzy environment. *Fuzzy Sets and Systems*, 81(1):103 – 122, 1996.

[DP95] Didier Dubois and Henri Prade. Possibility theory as a basis for qualitative decision theory. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, (IJCAI 1995)*, 1995.

[DPT13] Didier Dubois, Henri Prade, and Fayçal Touazi. Conditional preference nets and possibilistic logic. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, (ECSQARU 2013), Utrecht, The Netherlands*, 2013.

[DR03] Rina Dechter and Irina Rish. Mini-buckets: A general scheme for bounded inference. *Journal of the ACM*, 50(2):107–153, 2003.

[DT99] Jon Doyle and Richmond H. Thomason. Background to qualitative decision theory. *AI Magazine*, 20, 1999.

[Ehr96] Matthias Ehrgott. A characterization of lexicographic max-ordering solutions. Technical report, University of Kaiserslautern, Department of Mathematics, 1996.

[Ehr05] Matthias Ehrgott. *Multicriteria Optimization (2. ed.)*. Springer, 2005.

[FHWW10] Eugene C. Freuder, Robert Heffernan, Richard J. Wallace, and Nic Wilson. Lexicographically ordered constraint satisfaction problems. *Constraints*, 15(1):1–28, 2010.

[FL93] Hélène Fargier and Jérôme Lang. Uncertainty in constraint satisfaction problems: A probabilistic approach. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*. Springer Berlin Heidelberg, 1993.

[FLS93] Hélène Fargier, Jérôme Lang, and Thomas Schiex. Selecting preferred solutions in Fuzzy Constraint Satisfaction Problems. In *Proceedings of the 1st European Congress on Fuzzy and Intelligent Technologies*, 1993.

[FLS96] Hélène Fargier, Jérôme Lang, and Thomas Schiex. Mixed constraint satisfaction: A framework for decision problems under incomplete

knowledge. In *Proceedings of the 13th National Conference on Artificial Intelligence, (AAAI 1996), Portland, Oregon*, 1996.

[FP99] Hélène Fargier and Patrice Perny. Qualitative models for decision under uncertainty without the commensurability assumption. In *Proceedings of the 15th conference on Uncertainty in Artificial Intelligence, (UAI 1999)*, 1999.

[FRW10] Hélène Fargier, Emma Rollon, and Nic Wilson. Enabling local computation for partially ordered preferences. *Constraints*, 15(4):516–539, 2010.

[FW92] Eugene C. Freuder and Richard J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58(1-3):21–70, 1992.

[Gav01] Marco Gavanelli. Partially ordered constraint optimization problems. In *Proceedings of 7th International Conference on Principles and Practice of Constraint Programming, (CP 2001)*, 2001.

[Gav02] Marco Gavanelli. An implementation of Pareto optimality in CLP(FD). In *CP-AI-OR - International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems*, 2002.

[GMP+01] Ian P. Gent, Ewan MacIntyre, Patrick Prosser, Barbara M. Smith, and Toby Walsh. Random constraint satisfaction: Flaws and structure. *Constraints*, 6(4):345–372, 2001.

[GPD11] Christophe Gonzales, Patrice Perny, and Jean-Philippe Dubus. Decision making with multiple objectives using GAI networks. *Artificial Intelligence*, 175:1153–1179, 2011.

[GPR+10a] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K. Brent Venable, and Toby Walsh. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence*, 174(3-4):270–294, 2010.

[GPR+10b] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K. Brent Venable, and Nic Wilson. Interval-valued soft constraint problems. *Annals of Mathematics and Artificial Intelligence*, 58:261–298, April 2010.

[HK95] Robin Miles Hogarth and Howard Kunreuther. Decision making under ignorance: Arguing with yourself. *Journal of Risk and Uncertainty*, 10(1):15–36, 1995.

[Hun07] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, May-Jun 2007.

[Jun04] Ulrich Junker. Preference-based search and multi-criteria optimization. *Annals of Operations Research*, 130:75–115, 2004.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[KP08] Souhila Kaci and Henri Prade. Mastering the processing of preferences by using symbolic priorities in possibilistic logic. In *Proceedings of the 18th European Conference on Artificial Intelligence, (ECAI 2008)*, 2008.

[KR76] Ralph L. Keeney and Howard Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[Kum92] Vipin Kumar. Algorithms for constraint satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[Lar92] Oleg I. Larichev. Cognitive validity in design of decision-aiding techniques. *Journal of Multi-Criteria Decision Analysis*, 1(3):127–138, 1992.

[LKM10] Ramzi Ben Larbi, Sébastien Konieczny, and Pierre Marquis. A characterization of optimality criteria for decision making under complete ignorance. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning, (KR 2010), Toronto, Canada*, 2010.

[LM95] Oleg I. Larichev and Helen M. Moshkovich. ZAPROS-LM – a method and system for ordering multiattribute alternatives. *European Journal of Operational Research*, 82(3):503–521, 1995.

[LMS99] Javier Larrosa, Pedro Meseguer, and Thomas Schiex. Maintaining Reversible DAC for Max-CSP. *Artificial Intelligence*, 107:149–163, 1999.

[LS82] Graham Loomes and Robert Sugden. Regret theory: An alternative theory of rational choice under uncertainty. *The Economic Journal*, 92(368):805–824, 1982.

[Mac77] Alan K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Mar11] Radu Marinescu. Efficient approximation algorithms for multi-objective constraint optimization. In *Proceedings of the 2nd International Conference on Algorithmic Decision Theory, (ADT 2011)*, 2011.

[Mas79] Eric Maskin. Decision-making under ignorance with implications for social choice. *Theory and Decision*, 11:319–337, 1979.

[MMO02] Helen M. Moshkovich, Alexander I. Mechitov, and David L. Olson. Ordinal judgments in multiattribute decision analysis. *European Journal of Operational Research*, 137(3):625–641, 2002.

[MRW12] Radu Marinescu, Abdul Razak, and Nic Wilson. Multi-objective influence diagrams. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence, (UAI 2012), Catalina Island, CA, USA*, 2012.

[MRW13] Radu Marinescu, Abdul Razak, and Nic Wilson. Multi-objective constraint optimization with tradeoffs. In *Principles and Practice of Constraint Programming*. Springer Berlin Heidelberg, 2013.

[OW11] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance: an extension to Pareto Dominance and its application in Soft Constraints. In *Proceedings of CP 2011 Workshop on Preferences and Soft Constraints (CP 2011)*, 2011.

[OW12] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance: an extension to Pareto Dominance and its application in Soft Constraints. In *Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence, (ICTAI 2012)*, 2012.

[OW13] Conor O'Mahony and Nic Wilson. Sorted-Pareto Dominance and qualitative notions of optimality. In *Proceedings of the 12th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, (ECSQARU 2013)*, 2013.

[Par97] Vilfredo Pareto. *Cours d'économie politique*. Rouge, Lausanne, 1897.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Fran., CA, USA, 1988.

[PS05] Patrice Perny and Olivier Spanjaard. A preference-based approach to spanning trees and shortest paths problems. *European Journal of Operational Research*, 162:584–601, 2005.

[PSS06] Patrice Perny, Olivier Spanjaard, and Louis-Xavier Storme. A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147:317–341, 2006.

[Raw71] John Rawls. *A Theory of Justice*. Oxford University Press, New York, 1971.

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*. Elsevier Science Inc., New York, NY, USA, 2006.

[Sav51] Leonard J. Savage. The theory of statistical decision. *Journal of the American Statistical Association*, 46:55–67, 1951.

[Sav54] Leonard J. Savage. *The Foundations of Statistics*. Wiley, 1954.

[Scha] Thomas Schiex. Cost Function : A constraint programming extension. http://www.costfunction.org. Accessed July 2013.

[Schb] Thomas Schiex. The CELAR Radio Link Frequency Assignment Problems. http://www.inra.fr/mia/T/schiex/Doc/CELAR.shtml. Accessed July 2013.

[Schc] Thomas Schiex. WCSP file format. http://costfunction.org/mobyle/htdocs/portal/help/wcsp.html. Accessed July 2013.

[Sch00] Thomas Schiex. Arc consistency for soft constraints. In *Proceedings of the 6th International Conference on the Principles and Practice of Constraint Programming, (CP 2000)*, 2000.

[SD96] Barbara M. Smith and Martin E. Dyer. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence*, 81:155 – 181, 1996.

[Sen70] Amartya K. Sen. *Collective choice and social welfare*. North-Holland Publishing Co., Amsterdam, 1970.

[SFV95] Thomas Schiex, Hélène Fargier, and Gérard Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the*

14th International Joint Conference on Artificial Intelligence, (IJCAI 1995), 1995.

[SH81] Linda G. Shapiro and Robert M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:504–519, 1981.

[Sho83] Anthony F. Shorrocks. Ranking income distributions. *Economica*, 50(197):3–17, February 1983.

[TF02] Marc Torrens and Boi Faltings. Using soft CSPs for approximating Pareto-optimal solution sets. In *Proceedings of AAAI 2002 Workshop: Preferences in AI and CP: Symbolic Approaches*, 2002.

[TSS09] Stefan Traub, Christian Seidl, and Ulrich Schmidt. An experimental study on individual choice, social welfare, and social preferences. *European Economic Review*, 53(4):385–400, 2009.

[VLS96] Gérard Verfaillie, Michel Lemaître, and Thomas Schiex. Russian doll search for solving constraint optimization problems. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI 1996) and 8th. Conference on Innovative Applications of Artificial Intelligence (IAAI 1996)*, 1996.

[vN28] John von Neumann. On the theory of games of strategy. *Mathematische Annalen*, 100:295–320, 1928.

[vNM47] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[Wal50] Abraham Wald. *Statistical Decision Functions*. John Wiley, NY., 1950.

[Wal94] Richard J Wallace. Directed arc consistency preprocessing as a strategy for maximal constraint satisfaction. In *Working Notes of the ECAI 1994 Workshop on Constraint Processing, (ECAI 1994)*, 1994.

[Wal02] Toby Walsh. Stochastic constraint programming. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, (ECAI 2002)*, 2002.

[WF08] Nic Wilson and Hélène Fargier. Branch-and-bound for soft constraints based on partially ordered degrees of preference. In *Proceedings of ECAI 2008 Workshop on inference methods based on graphical structures of knowledge, (WIGSK 2008)*, 2008.

[WO11] Nic Wilson and Conor O'Mahony. The relationships between qualitative notions of optimality for decision making under logical uncertainty. In *Proceedings of the 22nd Irish Conference on Artificial Intelligence and Cognitive Science, (AICS 2011)*, 2011.

[WT11] Nic Wilson and Walid Trabelsi. Pruning rules for constrained optimisation for conditional preferences. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming, (CP 2011)*, 2011.

[Yag88] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.